

NOVEMBER 1988

COMPUTER



THE PROGRAMMER'S APPRENTICE

Companion article to
November *IEEE Software* and
Winter *IEEE Expert* issues on
expert-system software

Also in this issue:

- ☐ Contention Reduction
- ☐ Program Correctness
- ☐ Architecture Taxonomy



IEEE COMPUTER SOCIETY



THE INSTITUTE OF ELECTRICAL AND
ELECTRONICS ENGINEERS, INC.



GET ON THE "PAWS™ EXPRESSWAY"



TO SYSTEM DESIGN SOLUTIONS

Why not take the shortest and fastest route toward system design solutions? To define and model a large, complex system, you need a modeling tool that can address the system's total architecture.



PAWS/GPSM™ is an easy-to-use high productivity modeling and simulation tool that lets you evaluate alternative architectures while focusing on performance. It provides a state-of-the-art environment for developing accurate and reliable product definitions.



PAWS provides high level primitives closely resembling the user's intuition. PAWS models the behavior of a total system implemented in diverse technologies including software, hardware, and firmware.



GPSM, the graphical interface to PAWS, helps you quickly and easily transfer your ideas into pictures. GPSM helps you visualize multiple data and control flows through several components and incrementally synthesize models of a total system.



PAWS/GPSM is flexible. It lets you refine your model as your system design evolves so you can eliminate potential problems as early as possible in the product design cycle.

Call (512) 474-4526 to receive information on PAWS/GPSM. And get on the PAWS expressway to system design solutions.

PAWS and GPSM are trademarks of Information Research Associates, Inc.

IRA

Information Research Associates • 911 W. 29th St. • Austin, Texas 78705 • (512) 474-4526

Reader Service Number 1

COMPUTER

November 1988

Published by the IEEE Computer Society

Vol. 21, No. 11

ARTICLES

10 The Programmer's Apprentice: A Research Overview

Charles Rich and Richard C. Waters

How can we keep programmers from wasting time and expertise on routine tasks? By providing them with an intelligent computer program that functions like a human support team.

26 Reducing Contention in Shared-Memory Multiprocessors

Per Stenström

Contention for shared resources such as memory and an interconnection network limit the performance of shared-memory multiprocessors. The implementations reviewed here demonstrate ways to reduce contention.

38 Using Inspections to Investigate Program Correctness

Robert N. Brier

An approach to software inspections that examines the properties underlying correct programs would help programmers verify correctness and build it into their programs.

46 A Taxonomy for Computer Architectures

David B. Skillicorn

This taxonomy extends Flynn's classification of architectures to be more discriminating. In particular, the growing variety of multiprocessors can be categorized and related.

COMPUTER

10662 Los Vaqueros Circle, Los Alamitos, CA 90720

Editor-in-Chief

Bruce D. Shriver, University of Hawaii

Editorial Board

Dharma P. Agrawal, North Carolina State University

Dennis R. Allison, Stanford University

Scott Davidson, AT&T

Richard Eckhouse, MOCO

Scott E. Fahlman, Carnegie Mellon University

Gideon Frieder, Syracuse University

Edmund Gullizzi, Eckerd College

John Gurd, University of Manchester

James H. Hayes, University of California, Santa Cruz

Wiley McKinzie, Rochester Institute of Technology

K.M. Mohiuddin, IBM Almaden Research Laboratory

Yale Patt, University of California, Berkeley

William Riddle, Software Productivity Consortium

Sallie Sheppard, Texas A&M University

Helen M. Wood, National Oceanic and Atmospheric Administration

Akinori Yonezawa, Tokyo Institute of Technology

Submissions

Send six double-spaced copies of articles and special-issue proposals to Bruce D. Shriver, Dept. of Decision Sciences, University of Hawaii, 2404 Maile Way, Honolulu, HI 96822, phone (808) 948-7231. For electronic submission, Shriver's addresses are b.shriver on Compmail+, shriver@uhccuc on Bitnet, and shriver@uhccuc.uhcc.hawaii.edu.

Staff

Editor and Publisher: Tracie Seaborn

Assistant Publisher: Douglas L. Combs

Assistant to the Publisher: Pat Paulsen

Managing Editor: Marilyn Potes

Assistant Editors: Nancy Hays, Chuck Governale,

Steve H. Wilcox

Art Director: Jay Simpson

Design/Production: Miriam Wiegel

Membership Manager: Christina Champion

Circulation Manager: Paul Zive

Advertising Director: Dawn Peck

Advertising Coordinators: Heidi Rex, Marian Tibayan

Magazine Advisory Committee

Sallie Sheppard (chair), Sumit DasGupta, Michael Evangelist,

James J. Farrell III, Ted Lewis, David Pessel, Tracie Seaborn, Bruce D.

Shriver, John Staudhammer

Publications Board

James H. Aylor (chair), Victor Basili, J. Richard Burke, Jon T. Butler,

J.T. Cain, Sumit DasGupta, Michael Evangelist, James J. Farrell III,

Glen G. Langdon, Ted Lewis, Ming T. Liu, Theo Pavlidis, David Pessel,

Sallie Sheppard, Bruce D. Shriver, Charles B. Silio, John Staudhammer,

Harold Stone, Steven L. Tanimoto

DEPARTMENTS

- 4 President's Message**
Educational activities
- 8 Editor-in-Chief's Message**
AI and software technology: The linkages
- 9 Letters to the Editor**
- 60 Standards**
European conformance testing program;
NBS—now NIST—role expands with name change
- 64 Computer Society News**
Immigration reform concerns IEEE;
CSAB accredits 17 four-year CS programs
- 65 Update**
- 68 New Product Reviews**
Exploring expert systems—ESIE, Exsys, and VP-Expert;
PC-based shells for first-time developers—Clips and Personal Consultant Plus
- 82 New Products**
- 88 IC Announcements**
- 89 Microsystem Announcements**
- 91 Conferences**
Neural networks; WorldMed 88
- 95 Calendar**
- 101 Call for Papers**
- 115 Book Reviews**
Database Design Fundamentals; *Steve Jobs*; *Database Computers*; *Speech Communication*



Programming often seems less science than sorcery, where modern-day Merlins conjure up their own assistants, as in "The Programmer's Apprentice: A Research Overview." (See p. 10.)

Our cover wizard, seen with a unicorn before airbrushing and expert-system altering, is by Bernard Hugill, Michael Woodward Licensing, West Yorkshire, England. Cover design and production are by Jay Simpson and Jim Brummond, Design & Direction.

In the next issue

Mapping the Design Information Representation; A Case for Direct-Mapped Caches; SEGUE—Support for Distributed Graphical Interfaces; Inheritance of Synchronization and Recovery Properties in Avalon/C++; Software for Supercomputers

Computer Society Information Page, inside back cover; Membership Application, 103; Change-of-Address Form, 44; Career Opportunities, 104; Advertiser/Product Index, 120; Reader Service Card, 120A

Circulation: *Computer* (ISSN 0018-9162) is published monthly by the IEEE Computer Society: IEEE Headquarters, 345 East 47th St., New York, NY 10017; Computer Society Publications Office, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720, phone (714) 821-8380. Annual subscription included in society member dues. Nonmember prices: available upon request. Single copy prices: members \$10.00; nonmembers \$20.00. This magazine is also available in microfiche form.

Postmaster: Send undelivered copies and address changes to *Computer*, IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854. Second class postage is paid at New York, New York, and at additional mailing offices.

Copyright and reprint permission: Copyright © 1988 by the Institute of Electrical and Electronics Engineers, Inc. All rights reserved. Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limits of US copyright law for private use of patrons: (1) those post-1977 articles that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 29 Congress Street, Salem, MA 01970; (2) pre-1978 articles without fee. Instructors are permitted to photocopy isolated articles for noncommercial classroom use without fee. For other copying, reprint, or republication permission, write to True Seaborn, Editor and Publisher, *Computer*, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720.

Editorial: Unless otherwise stated, bylined articles, as well as product and service descriptions, reflect the author's or firm's opinion. Inclusion in *Computer* does not necessarily constitute endorsement by the IEEE or the Computer Society. All submissions are subject to editing for style, clarity, and space considerations. *Computer* subscribes to the Computer Press Association's code of professional ethics.

BPA

Computer Society educational activities emphasize accreditation, curriculum development



Edward A. Parrish, Jr.

This month I asked Gerald Engel, vice president for educational activities, for an update on the programs within his area of responsibility. The education of computer science and engineering professionals has always been of great concern to the Computer Society, and much has been accomplished recently, especially through our accreditation and curriculum development programs.

Edward A. Parrish, Jr.
President

It is indeed a pleasure to report on the Computer Society's educational activities. This year, I've had the honor of working with the many society volunteers who are involved in these programs. The success we've had is a tribute to their dedication and to that of the Computer Society staff. Both groups have my heartfelt thanks.

Like other society activities, the programs of the Educational Activities Board were audited and reviewed in 1987. The review suggested giving the highest priority to accreditation and curriculum development projects. It also recommended undertaking, where appropriate, cooperative activities with our sister societies. We have been very successful in both these regards.

Accreditation. The society's accreditation activities involve both the Accreditation Board for Engineering and Technology (ABET) and the Computing Sciences Accreditation Board (CSAB). Our involvement with ABET is through the IEEE, and we provide significant input on program criteria in computer-related areas. As of July 1988, ABET's Engineering Accreditation Commission had 49 accredited programs in the "computer group." The Technology Accreditation Commission lists 19 accredited associate degree programs and 13 accredited bachelor degree programs within the "computer group." Suggestions regarding appropriate criteria in both the engineering and technology programs are welcome, as are suggestions for individuals to serve as program evaluators in the computer area.

ABET is in its 57th year, but CSAB just completed its third round of accreditation actions. The Computer Society and ACM formed CSAB to fill a gap in program quality control caused by ABET's restriction to programs with



Gerald L. Engel

"engineering" in their title. Obviously, many computer science programs do not meet that requirement. As of July 1988, CSAB's Computer Science Accreditation Commission had accredited 65 computer science programs. As with ABET, input regarding program criteria and nominations for program evaluators are strongly encouraged.

As a new organization in accreditation activities, CSAB has been working to achieve appropriate recognition of its role. In a letter dated August 29, 1988, William J. Bennett, then secretary of the United States Department of Education, indicated that he is to "list the Computer Science Accreditation Commission of the Computing Sciences Accreditation Board as a nationally recognized agency for the accreditation

of baccalaureate programs in computer science." With this recognition achieved, CSAB is now seeking recognition by the Council on Post-secondary Accreditation (COPA). Currently, CSAB holds applicant status with COPA, and it is hoped that final recognition can be achieved soon. A hearing regarding the application for recognition is scheduled for January, and it is important that all opinions regarding computer science program accreditation and CSAB's approach be heard. Should you have feelings on this matter, please communicate them as soon as possible to

T. Phillips
Computing Sciences Accreditation
Board
35 East 47th Street
New York, NY 10017

Looking to the future in the accreditation area, some interest has been expressed in undertaking program accreditation for information systems. Draft criteria have been prepared, and discussions are now going on with sister societies and accreditation organizations with an interest in this area.

Curriculum. The society, in conjunction with ACM, has begun a major project to develop modern, updated curriculum recommendations in computer science and engineering. The Computer Society's 1983 Model Program was becoming dated, as were ACM's 1978 recommendations. Considering the field's current status and maturity, both organizations felt that a single, coordinated effort would best serve everyone's interest. Thus, a task force consisting of five representatives of each society is currently at work. An initial report on this effort is expected during Compeon Spring 89. As the project develops, individuals will be needed both to review materials and to provide input to the task force. Interested parties should contact me.

The other current curriculum project deals with laboratories. The purpose of this project is to develop a series of monographs detailing all aspects of each laboratory needed for the computer science and engineering curriculum. The monographs will address such issues as necessary support, equipment, and costs, and will include recommended exercises and projects. The first

of these monographs should be available from the Computer Society Press around the end of the year.

New award. The accreditation and curriculum projects encourage and recognize quality at the program level. To recognize excellence in education at the individual level, the Computer Society has established the annual Taylor L. Booth Education Award. This award, approved by IEEE last summer, is named for our late distinguished colleague and former vice president for educational activities. Its purpose is "to recognize individuals for their outstanding record in computer science and engineering education." Additional information on this award and a solicitation for nominations is scheduled to appear shortly.

Other projects. A variety of additional joint projects exist. Within the Computer Society, we are exploring ways of cooperating with the Technical Committee on Computers and Education, and with the Area Activities Board, regarding student chapters.

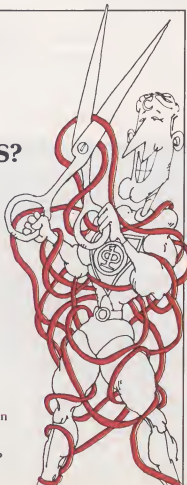
Within the IEEE, we are cooperating on a brochure to present the computer science and engineering field to junior high and high school students. We also cooperate on ABET accreditation, as described above, and on ways to better integrate the computer in electrical engineering education. The latter topic may well lead to a more general project addressing service courses and jointly administered curricula.

With ACM, we recently completed a project addressing requirements for teachers of computer science at the high school level. Finally, with ACM and the Mathematical Association of America, we completed a project considering the special problems and opportunities that exist when computer science is offered within a mathematics department.

LATE MAGAZINES? NO MAGAZINES? MEMBERSHIP STATUS PROBLEMS? NO ANSWERS TO YOUR COMPLAINTS?

*Let your
Computer Society
Ombudsman cut
through the red
tape for you.*

IEEE COMPUTER SOCIETY Ombudsman
10662 Los Vaqueros Circle
Los Alamitos, CA 90720
(714) 821-8380 / COMPMAIL+: CS.HELP



I have tried to briefly summarize some of the activities of the Educational Activities Board. Where you have an interest in these activities, please let us know. Should you have ideas for new activities, please suggest them. The board will meet later this month in conjunction with Supercomputing 88 in Kissimmee, Florida, and all interested parties are encouraged to attend. Whether or not you are able to attend the meeting, please contact us with your suggestions and ideas.

Gerald L. Engel
Vice president for education

MULTIPLY THE BENEFITS OF CASE

with the LAN BASED VISIBLE SOLUTION®

Coherent project development builds high quality systems faster!!!

All project team members use the same data dictionary!!!

Methodology based expert rules checks all work for consistency across the entire project!!!

Flexible, friendly, easy-to-use graphical interface brings project staff up-to-speed quickly!!!

VISIBLE ANALYST® WORKBENCH

THE NETWORKABLE VERSION 3.0--NEW FEATURES

- Identical look & feel as single user version • Configurable executable modules •
- Unlimited number of potential users on the LAN • Little or no system degradation resulting from Workbench use • Project wide rules validation • Multiple concurrent users • Common data dictionary • Full file and record locking • Multiple security levels •

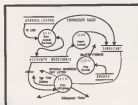
Ever since computer aided software engineering products came on the market, interest has been high. And so, not surprisingly, has been the cost of owning them. But now Visible Systems is offering state of the art CASE technology at an incredibly affordable price \$595.00 per module. These modular tools give your company the ability to grow in CASE power. The modules of the Workbench build upon each other to provide more and more productivity.

VISIBLE ANALYST - MODULE A

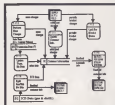
The Visible Analyst is the foundation of the Workbench. The user can create any analysis or design diagram. Multiple line styles, text fonts, and symbol types are standard features. Other features include a custom symbol generator and a reusable constructs library. Build and store entire diagrams for reuse in other projects. Boilerplates allow diagrams to be customized. Diagrams can be output in presentation quality using HP laser printers. Entire subsystems or single diagrams can be moved from one project to another while maintaining nested relationships. The LAN version allows multiple analysts to build or edit diagrams, within the same project, concurrently!!!

VISIBLE RULES - MODULE B

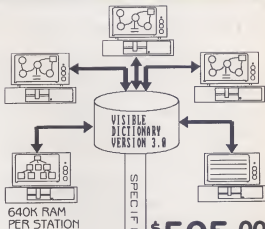
The Visible Rules is the on-line expert system that makes the use of structured techniques possible. The rules module implements either of two powerful, methodology based, expert rules systems. Yourdon/DeMarco or Gane & Sarson. Once enabled, the rules system monitors the diagramming process for naming conventions, data balancing, proper symbol usage etc. After diagrams have been built, the rules module analyzes them for errors. Error reports are output to screen or hardcopy. And the analysis takes place on-line. One never leaves the diagram. Problem resolution and identification is faster and easier. Logical data flow splitting is fully supported. Dictionary is automatically populated. Unlimited levels of nested decompositions are allowed.



YOURDON DIAGRAM



GANE & SARSON



\$595.00
(PER MODULE)

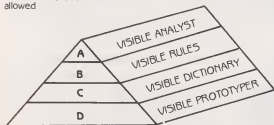
IBM PC BASED
NOVELL NETWORK
LAN SUPPORT

VISIBLE DICTIONARY - MODULE C

Visible Dictionary is the central data repository for the system being modeled. It is multi-user and interactive. Many users may access it simultaneously. Full file and record locking capabilities keep users from colliding and corrupting data. Visible Dictionary has many powerful features including "where used" listings, wild card searches and finds, full import/export, automatic entry generation and entry updating, on-line accessibility from diagrams, many different entry types, customized report formatting, and much more. The Dictionary organizes, reports, and categorizes all project data. File format information is provided for data transfers in ASCII format. The specification developed using the Visible Workbench becomes a company asset that can be used to build upon in later stages of the systems development lifecycle.

NEW!!! VISIBLE PROTOTYPING - MODULE D

The Visible Prototyper is our new screen design and full system simulation tool. Four levels of prototyping are supported: Screen design and panel linking, Panel branching, Data selection, and System simulation. User interface is "pre-approved" integrates with all design methodologies, use for education and training reduces costs. The Prototyper is available as a standalone or integrated fourth module with links to the Visible Dictionary for consistency in data names.



Visible Systems

CORPORATION

49 Lexington Street • Newton, MA 02165 • (617) 969-4100
FAX (617) 969-9505 Telex 261102 VSCUR

LEADING EDGE CASE TECHNOLOGY

SAME DAY SHIPMENTS WORLDWIDE

© 1988 Visible Systems Corporation

Visible, Visible Solution, The Visible Analyst and Visible Systems Corporation are registered Trademarks of Visible Systems Corporation

Reader Service Number 3

Editor-
in-Chief's

MESSAGE

AI and software technology: The linkages

Interest in the connections between artificial intelligence and software technology has increased markedly in the past few years. The literature now describes a wide range of both research and development activities that apply AI technology to the process of specifying, prototyping, developing, testing, and maintaining software.

About a year ago, I received a proposal from Murat Tanik and Raymond Yeh to do a special issue of *Computer* on this topic. I got together with Ted Lewis, editor-in-chief of *IEEE Software*, and David Pessel, editor-in-chief of *IEEE Expert*, and recommended they put together companion issues of their two magazines on this important topic. The recommendation was based on the excellent response we received on our first companion issues—the November 1987 issues of *Computer* and

IEEE Software on integrated design and programming environments. In my editor-in-chief's message for that issue, I had already introduced the idea of additional issues featuring complementary material and mentioned AI technology in software engineering as a possible topic. With such a proposal now in hand, Lewis and Pessel agreed that it presented an opportunity for providing the readership with significant, in-depth, and timely coverage of ongoing work in this area.

I'm publishing the table of contents of both *IEEE Software* and *IEEE Expert* here in *Computer* (see p. 16). I encourage those of you who do not receive either one or both of these magazines to pick up a copy and enjoy the results of the combined labors of Guest Editors Tanik and Yeh, Editors-in-chief Lewis and Pessel, myself,

numerous authors, a very large group of referees, and the staffs that assisted each of the above. For example, while Tanik was overseas for a part of the summer, he was aided by one of his graduate students, Lawrence Leff. I would personally like to thank Leff for the effort he put forth.

For its part, *Computer* is publishing a general article in this topical area (see "The Programmer's Apprentice," pp. 10-25, by Charles Rich and Richard C. Waters). I hope it will pique your interest in pursuing the articles in both *IEEE Software* and *IEEE Expert*.

I look forward to hearing of your reaction to this newest of our complementary issue series.

Bruce D. Shriver
Editor-in-chief

We wrote the book Now we've



Excelsior

Get a free video from the
leader in CASE.

It's called "The Excelsior® Difference." And it's a frank, management-to-management discussion about why Excelsior from Index Technology is so different from other CASE solutions. And why Excelsior/RTS' superior capabilities—with Index Technology's implementation support—can make a difference in your organization.

In this short video, you'll learn how Index Technology has helped the leaders in aerospace, defense and engineering. And how we can help you

Excelsior is a registered trademark and the Index Technology logotype is a trademark of Index Technology Corporation.

LETTERS

MVL error: Q should be R

To the Editor:

"Multiple-Valued Logic: A Tutorial and Appreciation" by Kenneth C. Smith (April 1988 *Computer*, pp. 17-27) is an interesting article of particular value to the nonspecialist reader. Unfortunately, the section on storage techniques in MVL (p. 22) contains an obvious minor error or misprint.

The last sentence in the middle column on page 22 reads "For $x_R = \bar{x}_S$, $x_Q = \bar{x}_Q = x_S$ once C has fallen to zero." Since in multiple-valued logic the principle of designing complement circuits remains the same, the $\bar{x}_Q$ in the statement should be replaced by $\bar{x}_R$, which will make the notation $x_S = x_Q$ still valid for Figure 5 (also p. 22).

Yu-Chun Donald Lie
Taipei City, Taiwan

Yes, this is an error—of unknown origin (author Smith has misplaced the original manuscript) while the article was being reviewed and revised for publication. The sentence should read "For $x_R = \bar{x}_S$, $x_Q = \bar{x}_R = x_S$ once C has fallen to zero."

—Ed.

Computer welcomes your letters. Send technical correspondence to Bruce D. Shriver, Computer Editor-in-Chief, Dept. of Decision Sciences, University of Hawaii, 2404 Maile Way, Honolulu, HI 96822. Send other comments to Letters Editor, *Computer*, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720.

All submissions are subject to editing for style, length, and clarity.

Hyper-semantic data modeling

To the Editor:

Several readers of "Traditional, Semantic, and Hyper-Semantic Approaches to Data Modeling" (Potter and Trueblood, June 1988 *Computer*, pp. 53-63) have written to me requesting additional information on the hyper-semantic approach. Although the traditional and semantic approaches are well referenced in the article, there are no references for the reader to follow with respect to hyper-semantic modeling.

Interested readers are referred to "A Unified Approach to Modeling Knowledge and Data" by Potter and Kerschberg in the *Proceedings of the IFIP TC2 Working Conference on Knowledge and Data*. These proceedings will soon be available from North-Holland Publishing Company as the second installment in their Data Semantics Series.

Don Potter
University of Georgia

on CASE. made the movie

design, document and deliver better systems, on time and within budget.

You'll also learn why Index Technology is the smartest choice now. And why we'll continue to lead the industry for years to come.

Call for your free video, "The Excelerator Difference." And soon our story will be appearing in a post office box near you. So don't wait. Just call us toll free at 1-800-777-8858.

Free:

Index Technology

TM

Index Technology Corp.
One Main Street
Cambridge, MA 02142

Reader Service Number 4



The Programmer's Apprentice:



How can we keep programmers from wasting time and expertise on routine tasks? By providing them with an intelligent computer program that functions like a human support team.

A Research Overview

Charles Rich and Richard C. Waters
Massachusetts Institute of Technology

The long-term goal of the Programmer's Apprentice project is to develop a theory of how expert programmers analyze, synthesize, modify, explain, specify, verify, and document programs. This is basic research at the intersection of artificial intelligence and software engineering. From the AI perspective, we are using programming as a domain in which to study fundamental issues in knowledge representation and reasoning. From the software engineering perspective, we are applying techniques from AI to automate the programming process. Other work in this intersection area is reviewed in this month's issues of *IEEE Expert* and *IEEE Software*.

Because it will be a long time before we can fully duplicate human programming abilities, the near-term goal of the project is to develop a system—the Programmer's Apprentice—that will provide intelligent assistance in all phases of the programming task.

The goals of this article are to lay out our vision of the Programmer's Apprentice, the principles and techniques underlying it, and our progress toward it. The primary vehicle for this exposition is three scenarios illustrating the use of the Apprentice in three phases of the programming task: implementation, design, and requirements. The first scenario is taken from a completed working prototype. The second and third scenarios are the targets for prototype systems currently under construction.

Before presenting the scenarios, we discuss the two basic principles underlying our work—namely, the assistant approach and inspection methods (clichés)—and our two main technical advances: the Plan Calculus and a hybrid reasoning system (Cake).

The assistant approach

One approach to solving current software problems is to eliminate programmers through automatic programming. As typically conceived, automatic programming calls for an end user to write a complete specification for what is desired; a completely automatic system then generates a program satisfying this specification. Program generators of this type have been successfully developed for a number of narrow applications. However, completely automatic programming for a broad range of applications is not a realistic near-term goal.¹

A fundamental difficulty with the completely automatic approach is the trade-off between the generality of a specification language and its ease of use and compilation (implementation). Specification languages have been developed that are easy to use and relatively easy to compile, but only for narrow applications. General-purpose specification languages are much less satisfactory. Writing a complete specification in a general-purpose specification language (for example, first-order logic) is seldom easier—and often incredibly harder—than writing a program. Furthermore, there has been little success in developing automatic systems that compile efficient programs from such specifications.

An alternative approach is to assist programmers rather than replace them. A provocative example of the assistant approach was proposed by IBM's Harlan Mills in the early 1970s. He suggested creating "chief programmer teams" by surrounding expert programmers with support staffs of human assistants, including junior programmers, documentation writers, program librarians, and so on.

Productivity was thereby increased because the chief programmer could apply full effort to the most difficult parts of a given software task without getting bogged down in routine details that currently use up most of every programmer's time. Experience has shown that this division of labor can be very successful. Our goal is to provide every programmer with a support team in the form of an intelligent computer program called the Programmer's Apprentice.

We think of the Apprentice as a new agent in the software process (see Figure 1) rather than as a tool. It should interact with the programmer like a human assistant. Furthermore, both the programmer and the Apprentice have access to all the facilities of the existing programming environment. The key to making this approach work is communication between the programmer and the Apprentice: It must be based on a substantial body of shared knowledge of programming technique. If the programmer has to explain everything to the Apprentice from first principles, it would be easier to do the work alone.

The assistant approach lends itself to incremental progress. Initially, the Apprentice will be able to take over only the simplest and most routine parts of the programming task (as illustrated in the first scenario below). As technology advances, however, the portion of the task handled by the Apprentice will increase; we don't have to wait until some subtask can be totally automated. Note that some parts of a given program may always have to be hand-coded because of, for example, extremely strict performance requirements. However, there is always plenty of routine work to be done, especially in large systems.

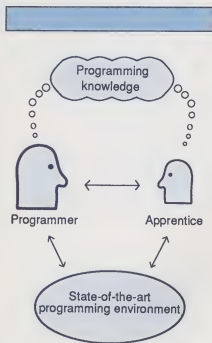


Figure 1. The programmer and the Apprentice are two cooperating agents in the software process. Both use tools in the programming environment through the existing interfaces. The scenarios in this article illustrate how the programmer and the Apprentice communicate with each other, and the shared programming knowledge that makes this communication possible.

Inspection methods

Human programmers seldom think only in terms of primitive elements, such as assignments and tests. Rather, like engineers in other disciplines, they think mostly in terms of commonly used combinations of elements with familiar names. We call these familiar combinations *clichés*. Successive approximation, device driver, and information system are examples of programming clichés spanning the range from low-level implementation ideas to high-level specification concepts. (For evidence of the psychological reality of programming clichés, see Soloway and Ehrlich.²)

In general, a cliché consists of roles and constraints. The roles of a cliché are the parts that vary from one occurrence of the cliché to the next. The constraints are used to specify fixed elements of structure (parts present in every occurrence), to verify that the parts that fill the roles in a particular

occurrence are consistent, and to compute how to fill empty roles in a partially specified occurrence of a cliché.

An essential property of clichés is their relationship to one another. For example, a cliché may be a special case or an extension of another cliché. Algorithmic and data structure clichés may be related as possible implementations of specification clichés.

Given a library of clichés, it is possible to perform many programming tasks by inspection rather than by reasoning from first principles. For example, in analysis by inspection, properties of a program are deduced by recognizing occurrences of clichés and referring to their known properties. In synthesis by inspection, implementation decisions are made by recognizing clichés in specifications and then choosing among various implementation clichés. (An intelligent assistant that uses a library of clichés about the *process* of programming is described by Huff and Lesser.³)

The Programmer's Apprentice focuses on the use of inspection methods to automate programming, as opposed to more general but harder to control methods, such as deductive synthesis or program transformations. In human programming, inspection methods are the most effective approach whenever applicable. However, since inspection methods are based ultimately on experience, they apply only to the routine parts of programming problems. This is compatible with the intended division of labor between the programmer and the Apprentice.

Codifying clichés is a central activity in the Programmer's Apprentice project. The scenarios below show parts of the libraries we are building in the areas of program implementation, design, and requirements. The scenarios also illustrate how a shared vocabulary of clichés serves as the language of communication between the programmer and the Apprentice.

The Plan Calculus

Clichés and inspection methods are theoretical (and perhaps psychological) concepts. To apply these ideas, clichés need to be represented in a concrete, machine-usable form. A cornerstone of the Programmer's Apprentice is a formal representation for programs and programming clichés called the *Plan Calculus*. This formalism was developed early in the proj-

ect; we will summarize only its key properties here (more details are available elsewhere⁴).

To a first approximation, the Plan Calculus can be thought of as combining the representation properties of flowcharts, dataflow schemas, and abstract data types. A plan is essentially a hierarchical graph structure made up of different kinds of boxes (denoting operations and tests) and arrows (denoting control and data flow). The representation has both a graphical notation (see the Plan Calculus example in the accompanying sidebar) and a formal semantics used for reasoning (see next section, "A hybrid reasoning system").

The Plan Calculus gives the Apprentice a "mental language" that abstracts away from the details of algorithms and data structures that are a result only of their expression in a particular programming language. For example, dataflow between two operations can be achieved either by setting a variable and then using it, or (if the language allows) by nesting the invocation of the producing operation inside the invocation of the consuming operation. Similarly, the same net control flow between operations and tests can be achieved by many different combinations of conditional primitives available in different languages. The explicit representation of control and dataflow in the Plan Calculus greatly simplifies all of the manipulations that need to be performed to support the programming task, as well as making most of the internals of the Apprentice programming-language independent.

In the architecture of the Programmer's Apprentice, program text (in Lisp or Ada, for example) is considered only a *presentation* of program structure for communication between the programmer and the Apprentice, or between the Apprentice and other parts of the programming environment (for example, the compiler). Other presentations, such as graphics, may also be appropriate in certain circumstances, though we are not pursuing this research direction at the moment.

The Plan Calculus is also a wide-spectrum formalism. Each operation and test in a plan has associated with it a set of preconditions and postconditions specified in a logical language. A plan in which all of the operations and tests are specified to be the primitives of some programming language is equivalent to a concrete program. However, the Plan Calculus is also used to represent partially designed pro-

grams and programming clichés, in which case the operations and tests in a plan may have abstract or incomplete specifications, and there may be arbitrary logical constraints, as well as dataflow and control flow.

Taxonomic relationships between clichés—specialization, for example—are handled by special-purpose mechanisms in

the cliché library. The relationship between a specification and an implementation is represented in the Plan Calculus by an *overlay* (see the example in the sidebar). Formally, an overlay defines a mapping from the set of instances of the implementation plan to the set of instances of the specification. (This is a generalization of the *abstraction function* in the

abstract data type methodology.) A cliché library may contain different overlays with the same domain and/or range, corresponding to different ways of abstracting the same implementation and/or different ways of implementing the same specification.

A key feature of overlays is that they are used for both analysis and synthesis by

A Plan Calculus example

The accompanying diagram is an example of the graphical notation for an overlay in the Plan Calculus. The name of the overlay is "bump-and-update-as-push." In general, an overlay diagram has a plan diagram on each side, with a set of hooked lines, called *correspondences*, between them. The plan diagram on the left defines the implementation, which is the domain of the mapping defined by the overlay; the plan diagram on the right defines the specification, which is the range of the mapping; the correspondences define the mapping.

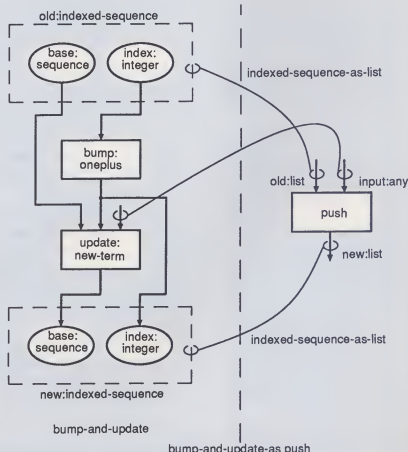
In this example the right side of the overlay is a degenerate plan diagram with only a single box; the specification being implemented in this overlay is the *push* operation on a *list*. (List is a data abstraction having a head of any type and a tail, which is a list or empty.) The diagram shows that push has two inputs, the old list and the input (of any type), and one output, the new list. The postcondition of push (logical specifications usually are not shown in diagrams) specifies that the head of the new list is equal to the input and that the tail of the new list is equal to the old list.

The left side of this overlay is a plan diagram representing a cliché combination of operations on an *indexed sequence* (a base sequence with an associated *index* integer), in which the index is decremented and a new term is stored. The name of this plan is "bump-and-update," and it has four roles: old (an indexed sequence), bump (an operation that adds one), update (an operation that stores a new term in a sequence), and new (an indexed sequence). Dataflow constraints between these roles are indicated by solid arrows. (Since this plan has no conditional structure, there are no control flow

arrows; control flow is indicated in plan diagrams by cross-hatched areas.)

There are three correspondences in this overlay; two are annotated with the name of another overlay called "indexed-sequence-as-list." This means that the old indexed sequence of bump-and-update, viewed as a list according to indexed-sequence-as-list, corresponds to the old input of push, and similarly for the new roles. Indexed-sequence-as-list is a data abstraction function defined as follows: The head of the

list corresponds to the term of the base indexed by the index; the tail of the list is recursively defined as the list implemented by the indexed sequence with the same base and one minus the index; the empty list corresponds to the indexed sequence with index zero. The third correspondence in the diagram indicates that the input to the update operation in bump-and-update (the input for the new term—the other two inputs are the sequence and the index) corresponds to the object being pushed onto the list.



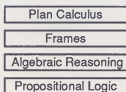


Figure 2. The hybrid knowledge representation and reasoning system (Cake) has a layered architecture.

inspection. The scenarios in this article concentrate on program synthesis; examples of the use of the Plan Calculus in program analysis can be found in a report by Wills.⁵

A hybrid reasoning system

The degree of automation the Apprentice can provide ultimately depends on its ability to reason about structured objects (programs, specifications, requirements) and their properties. Our approach to this reasoning task is to use a combination of special-purpose techniques and general-purpose logical reasoning.

Special-purpose representations and algorithms are essential to avoid the combinatorial explosions that typically occur in general-purpose logical reasoning systems. On the other hand, logic-based reasoning is very valuable when used, under tight control, as the "glue" between inferences made in different special-purpose representations.

This section describes a hybrid knowledge representation and reasoning system called Cake⁶ that we have developed and are using for all current work in the project. Figure 2 shows the architecture of Cake. Note that Cake combines special-purpose representations, such as frames and the Plan Calculus, with general-purpose logical and mathematical reasoning. Each layer of Cake builds on facilities provided by the layers below.

Figure 3, a short transcript from the currently running version of Cake, illustrates some of the facilities provided in the propositional, algebraic, and frame layers.

```

1> (Assertq P)
2> (Assertq (Implies P Q))
3> (Whyq Q)
  Q is True by Modus Ponens from:
    1. (Implies P Q) is True as a premise.
    2. P is True as a premise.
4> (Retractq P)
5> (Whyq Q)
  I don't know whether or not Q is true.
  (Assertq (And P (Not Q)))
6> **Contradiction: There is a conflict between the premises:
    1. (And P (Not Q)) is True.
    2. (Implies P Q) is True.
  Type cr to postpone dealing with this contradiction.
  Type premise number to retract one of the premises.
7> 1
  Retracting (And P (Not Q)) being True. . .
  #< Node (And P (Not Q)) : False>
8> (Assertq (= I J))
9> (Whyq (= (F I) (F J)))
  (= (F I) (F J)) is True by Equality from:
    1. (= I J) is True as a premise.
10> (Assertq (Transitive R))
11> (Assertq (R W X))
12> (Assertq (R X Y))
13> (Assertq (R Y Z))
14> (Whyq (R W Z))
  (R W Z) is True by Transitivity from:
    1. (R W X) is True as a premise.
    2. (R X Y) is True as a premise.
    3. (R Y Z) is True as a premise.
    4. (Transitive R) is True as a premise.
15> (Assertq (Subset A B))
16> (Assertq (Member X A))
17> (Whyq (Member X B))
  (Member X B) is True by Subsumption from:
    1. (Subset A B) is True as a premise.
    2. (Member X A) is True as a premise.
18> (Deftype Address (:Specializes Number))
19> (Deframe Interrupt
  (:Roles (Location Address) Program))
20> (Deframe Device
  (:Roles (Transmit Address) (Receive Address)))
21> (Deframe Interface
  (:Roles (Target Device) (From Interrupt) (To Interrupt))
  (:Constraints (= (Location ?From) (Receive ?Target))
    (= (Location ?To) (Transmit ?Target))))
22> (FInstantiate 'Interface :Name 'K7)
23> (FPut (>> 'K7 'Target 'Receive) 777777)
24> (FGet (>> 'K7 'From 'Location)
  777777)
25> (Why . . .)
  (= (Location (From K7))) is True by Equality from:
    1. (= (Location (From K7))
      (Receive (Target K7))) is True.
    2. (= (Receive (Target K7)) 777777) is True as a premise.
  
```

Figure 3. A verbatim transcript illustrating the reasoning capabilities of the propositional, algebraic, and frame layers of Cake.

(Line numbers in the following discussion refer to Figure 3.) These facilities are motivated in general by the desired characteristics of the Apprentice; we will later point out specific examples of their use in the scenarios.

The propositional layer of Cake provides three principal facilities. First, in lines 1-3 it automatically performs simple one-step deductions (technically, unit propositional resolution). Use of this very limited form of logical deduction is a kind of tight control essential to the hybrid architecture.

Second, the propositional layer acts as a recording medium for dependencies (often called a truth-maintenance system) and thus supports explanation (line 3) and retraction (lines 4-5). These facilities are motivated by the observation that when you delegate work to an assistant, you also need to have accountability and the ability to recover from mistakes in case it doesn't do what you expect.

Third, the propositional layer detects contradictions (lines 6-7). Furthermore, contradictions are represented explicitly in such a way that reasoning can continue with other information not involved in the contradiction. This feature is motivated by the desire for the Apprentice to support an evolutionary programming process. In this kind of process the programmer's knowledge is very often in an inconsistent state, particularly during the requirements acquisition and analysis phase.

The algebraic layer of Cake contains special-purpose decision procedures for congruence closure, common algebraic properties of operators (such as commutativity, associativity, and transitivity), partial functions, and the algebra of sets. The congruence closure algorithm in this layer determines whether or not terms are equal by substitution of equal subterms (lines 8-9). The decision procedure for transitivity (lines 10-14) determines when elements of a binary relation follow by transitivity from other elements. The algebra of sets (lines 15-17) involves the theory of membership, subset, union, intersection, and complements.

Equality reasoning is very important for the Apprentice because the formal semantics of the Plan Calculus makes heavy use of equality. Dataflow arrows in plans imply equalities between terms representing the source and destination points; correspondences in overlays are also equalities. Other algebraic properties—transitivity, commutativity, etc.—come up everywhere in the formal modeling of



Figure 4. Prototypes of the Programmer's Apprentice are being built by working inward from the two boundaries of the software development process.

data structures.

The frames layer of Cake supports the standard frame notions of inheritance ("Specializes" in line 18), slots ("Roles" in lines 19-21), and instantiation (line 22). The Apprentice's cliché library is organized on the basis of frame inheritance.

A notable feature of Cake's frame system, however, is that constraints are implemented in a general way. For example, the definition of the *interface* frame (line 21) has constraints between the roles of the instances filling its roles. When an instance of an interface is created (line 22) and a particular value ("777777") is put into one of its "second-level" roles (line 23), the same value can be retrieved from the other constrained role (line 24). This propagation is not achieved by ad hoc procedures but by the operation of the underlying logical reasoning system, including dependencies (line 25). Constraint propagation is intended to support the ability of the Apprentice to incrementally acquire information in any order.

The Plan Calculus layer of Cake supports graph-theoretic manipulations of plan and overlay diagrams, such as following arcs. It also implements the formal semantics of the Plan Calculus so that hybrid reasoning can take place involving both structure (as expressed by the diagrams) and function (as expressed in the preconditions, postconditions, and other logical annotations). In the semantics of the Plan Calculus, names of plans become predicate symbols, names of roles and overlays become function symbols, correspondences become equalities, dataflow becomes a combination of equalities and a partial order, and control flow becomes a combination of an equivalence relation and a partial order.

Programmer's Apprentice scenarios

Viewed most simply, the software development process has, at the beginning, the desires of an end user and, at the conclusion, a program that can be executed on a machine. The part of the software process closest to the user is typically called requirements acquisition; the part nearest the machine is typically called implementation; the middle of the process can be generally described as design. To achieve dramatic productivity improvements, the Programmer's Apprentice must eventually span the entire process. However, since this is a very large undertaking, we have begun by building prototypes of parts of the Apprentice.

Rather than try to define the boundaries between requirements acquisition, design, and implementation a priori (as, for example, in the traditional waterfall model), we have adopted the strategy of working inward from the two external boundaries, as Figure 4 shows. This strategy allows us to explore and discover the appropriate internal boundaries as we proceed. We want to emphasize, however, that this division into two efforts is a research strategy. We plan to eventually connect the two prototypes as a first demonstration of a complete Apprentice. Most likely we will then have to rebuild significantly on the basis of what we learn.

Three scenarios follow. The first two illustrate a progression of the Apprentice's capabilities in the areas of implementation and design; the third scenario illustrates capabilities in the requirements area. The first scenario is the actual output of a completed running system called KBEmacs. The second and third scenarios are target

Extending and complementing "The Programmer's Apprentice," two special issues on expert-system software:

November IEEE Software

A Tool to Coordinate Tools

Roberto Bisiani, François Lecouat, and Vincenzo Ambriola
Many development tasks are tedious and clerical. Tomorrow's environments will ease the burden, but what about today's environments like Unix? One aid is this planner.

Intelligent Support for Specifications Transformations

Jeffrey J.-P. Tsai and Joel C. Ridge
Because software development is knowledge-intensive, it makes sense for developers to use expert systems. The prototype specification-transformation system described here proves this.

Critiquing Software Specifications

Stephen Fickas and P. Nagarajan
Systems analysts bring a wealth of past experience with them. Their knowledge can be put to use in an automated critic for specification debugging.



Winter IEEE Expert

Knowledge-Based Support for Rapid Software Prototyping

Luqi
Interactive systems for computer-aided prototyping rely on reusable software components drawn from a software base.

KBRA: A New Paradigm for Requirements Engineering

Andrew J. Czuchry, Jr., and David R. Harris
This knowledge-based requirements assistant—part of Rome Air Development Center's knowledge-based software assistant—provides computer support for formal and informal requirements acquisition and analysis.

Interactive Problem Solving Using Task Configuration and Control

Elaine Kant
A task-representation-and-control system for incremental program construction is currently supporting research projects in program synthesis and mathematical-model design and understanding.



To order your copies, use the postcards on page 24A.

scenarios and are guiding work in progress on what we call the Design Apprentice and the Requirements Apprentice.

KBEmacs

KBEmacs (knowledge-based editor in Emacs) is a prototype of a part of the Programmer's Apprentice completed several years ago to demonstrate the utility of the assistant approach and clichés in the implementation part of the software process. Figure 5 shows how KBEmacs was used to implement a 54-line Ada program. (We have omitted the intermediate states of the editing session after each com-

mand. The full scenario is available elsewhere.⁷)

The implementation clichés used in this scenario are *simple report* (defined in the next section), *chain enumeration*, and *query user for key*. Note that *enumerator*, *main file key*, *title*, and *summary* are the names of roles in these clichés. Our current library of implementation clichés is similar in scope and level to earlier machine-usable codifications (see Green and Barstow,⁸ for example); however, as discussed above, we have developed an improved formal representation.

In essence, KBEmacs adds a new, higher level of editing commands to the existing text- and syntax-based commands of

Emacs (consistent with the assistant approach, the text- and syntax-based commands are still available). Using KBEmacs, changes in the algorithmic structure of a program can be achieved by a single command, even when they correspond to widespread textual changes in the program.

Two important capabilities of KBEmacs that Figure 5 does not illustrate but which appear in the full scenario⁷ are automatic generation of program documentation (explaining the program in terms of the clichés used) and programming-language independence. KBEmacs was originally constructed to operate on Lisp programs; relatively little effort was required to

Define a simple_report procedure UNIT_REPAIR_REPORT.
 Fill the enumerator with a chain_enumeration of UNITS and REPAIRS.
 Fill the main_file_key with a query_user_for_key of UNITS.
 Fill the title with ("Report of Repairs on Unit " & UNIT_KEY).
 Remove the summary.



```

with CALENDAR, FUNCTIONS, MAINTENANCE_FILES, TEXT_IO;
use CALENDAR, FUNCTIONS, MAINTENANCE_FILES, TEXT_IO;
procedure UNIT_REPAIR_REPORT is
  use DEFECT_IO, REPAIR_IO, UNIT_IO, INT_IO;
  CURRENT_DATE: constant STRING := FORMAT_DATE(CLOCK);
  DEFECT: DEFECT_TYPE;
  REPAIR: REPAIR_TYPE;
  REPAIR_INDEX: REPAIR_INDEX_TYPE;
  REPORT: TEXT_IO.FILE_TYPE;
  TITLE: STRING(1..33);
  UNIT: UNIT_TYPE;
  UNIT_KEY: UNIT_KEY_TYPE;
  procedure CLEAN_UP is
  begin
    SET_OUTPUT(STANDARD_OUTPUT);
    CLOSE(DEFECTS); CLOSE(REPAIRS); CLOSE(UNITS); CLOSE(REPORT);
  exception
    when STATUS_ERROR => return;
  end CLEAN_UP;
begin
  OPEN(DEFECTS, IN_FILE, DEFECTS_NAME); OPEN(REPAIRS, IN_FILE, REPAIRS_NAME);
  OPEN(UNITS, IN_FILE, UNITS_NAME); CREATE(REPORT, OUT_FILE, "report.txt");
  loop
    begin
      NEW_LINE; PUT("Enter UNIT Key: "); GET(UNIT_KEY);
      READ(UNITS, UNIT, UNIT_KEY);
      exit;
    exception
      when END_ERROR => PUT("Invalid UNIT Key"); NEW_LINE;
    end;
  end loop;
  TITLE := "Report of Repairs on Unit " & UNIT_KEY;
  SET_OUTPUT(REPORT);
  NEW_LINE(4); SET_COL(20); PUT(CURRENT_DATE);
  NEW_LINE(2); SET_COL(13); PUT(TITLE); NEW_LINE(60);
  READ(UNITS, UNIT, UNIT_KEY); REPAIR_INDEX := UNIT.REPAIR;
  while not NULL_INDEX(REPAIR_INDEX) loop
    READ(REPAIRS, REPAIR, REPAIR_INDEX);
    if LINE > 64 then
      NEW_PAGE; NEW_LINE; PUT("Page: "); PUT(INTEGER(PAGE - 1), 3);
      SET_COL(13); PUT(TITLE); SET_COL(61); PUT(CURRENT_DATE); NEW_LINE(2);
      PUT("  Date      Defect      Description/Comment"); NEW_LINE(2);
    end if;
    READ(DEFECTS, DEFECT, REPAIR.DEFECT);
    PUT(FORMAT_DATE(REPAIR.DATE)); SET_COL(13); PUT(REPAIR.DEFECT);
    SET_COL(20); PUT(DEFECT.NAME); NEW_LINE;
    SET_COL(22); PUT(REPAIR.COMMENT); NEW_LINE;
    REPAIR_INDEX := REPAIR.NEXT;
  end loop;
  CLEAN_UP;
exception
  when DEVICE_ERROR | END_ERROR | NAME_ERROR | STATUS_ERROR =>
    CLEAN_UP; PUT("Data Base Inconsistent");
  when others => CLEAN_UP; raise; end UNIT_REPAIR_REPORT;

```

Figure 5. A verbatim transcript from KBEmacs. Given the five commands above, it produced these 54 lines of Ada code.

```

with CALENDAR, FUNCTIONS, TEXT_IO;
use CALENDAR, FUNCTIONS, TEXT_IO;
cliché SIMPLE_REPORT is
  primary roles ENUMERATOR, PRINT_ITEM, SUMMARY;
  described roles FILE_NAME, TITLE, ENUMERATOR,
    COLUMN_HEADINGS, PRINT_ITEM, SUMMARY;
  comment "prints a report of {the input data of the enumerator}";
  constraints
    DEFAULT({the file_name}, "report.txt");
    DERIVED({the line_limit},
      66 - SIZE_IN_LINES({the print_item})
      - SIZE_IN_LINES({the summary}));
    DEFAULT({the print_item},
      CORRESPONDING_PRINTING({the enumerator}));
    DEFAULT({the column_headings},
      CORRESPONDING_HEADINGS({the print_item}));
  end constraints;
  use INT_IO;
  CURRENT_DATE: constant STRING := FORMAT_DATE(CLOCK);
  DATA: {};
  REPORT: TEXT_IO.FILE_TYPE;
  TITLE: STRING(1..{});
  procedure CLEAN_UP is
  begin
    SET_OUTPUT(STANDARD_OUTPUT); CLOSE(REPORT);
  exception when STATUS_ERROR => return;
  end CLEAN_UP;
begin
  CREATE(REPORT, OUT_FILE, {the file_name});
  DATA := {the input data of the enumerator};
  SET_OUTPUT(REPORT);
  TITLE := {the title};
  NEW_LINE(4); SET_COL(20); PUT(CURRENT_DATE); NEW_LINE(2);
  SET_COL(13); PUT(TITLE); NEW_LINE(60);
  while not {the empty_test of the enumerator}(DATA) loop
    if LINE > {the line_limit} then
      NEW_PAGE; NEW_LINE; PUT("Page: ");
      PUT(INTEGER(PAGE - 1), 3); SET_COL(13); PUT(TITLE);
      SET_COL(61); PUT(CURRENT_DATE); NEW_LINE(2);
      {the column_headings}({CURRENT_OUTPUT, modified});
    end if;
    {the print_item}({CURRENT_OUTPUT, modified},
      {the accessor of the enumerator}(DATA));
    DATA := {the step of the enumerator}(DATA);
  end loop;
  {the summary}({CURRENT_OUTPUT, modified});
  CLEAN_UP;
exception
  when DEVICE_ERROR | END_ERROR | NAME_ERROR
    | STATUS_ERROR =>
    CLEAN_UP; PUT("Data Base Inconsistent");
  when others => CLEAN_UP; raise;
end SIMPLE_REPORT;

```

Figure 6. An Ada presentation of the simple report cliché. This program text is analyzed by KBEmacs to extract the underlying plan, which is stored and can then be referred to in KBEmacs commands, as in Figure 5.

extend it to operate on Ada programs as well.

The major deficiencies of KBEmacs are related to the fact that it was completed before Cake was available. (In fact, many features of Cake are motivated in part by the deficiencies of KBEmacs.) The implementation of the Plan Calculus used in KBEmacs is essentially the graph formalism without the associated logical reasoning. This means, for example, that KBEmacs cannot reason about side effects in any general way. Also, constraints in KBEmacs (see the next section) are implemented by ad hoc procedures rather than by using a general-purpose scheme.

Defining an implementation cliché. Figure 6 illustrates a facility in KBEmacs for defining clichés using an extension of Ada syntax. KBEmacs automatically analyzes this text to extract the underlying plan for the cliché. KBEmacs commands can then be used to combine this plan with the plans for other clichés to make new programs. Figure 6 defines the simple-report cliché used in the scenario of Figure 5. The only extensions to the syntax of Ada in cliché definitions are the new defining form "cliché . . . is . . .," and the use of braces for role names and other annotation. Note, however, that this provides a more general form of parameterization than supported by the Ada package facility.

We envisage two levels of cliché definition activity using the Apprentice. First, this language-oriented interface makes it easy for a programmer to quickly define a cliché for (perhaps short-term) personal use. A second, much more intellectually demanding, task is to define a "suite" of clichés to be used by many people over a long period of time.

Like most clichés, simple report includes some standard computation (for example, the printing of the title page), some roles to be filled in (the title itself), and the dataflow and control flow between them. This cliché has seven roles:

- The *file name* is the name of the file that will contain the report being produced.
- The *title* is printed on a title page and (along with the page number) at the top of each succeeding page.
- The *enumerator* enumerates the elements of some aggregate data structure.
- The *print item* is used to print information about each of the enumerated elements.

- The *line limit* is used to determine when a page break should be inserted.
- The *column headings* are printed at the top of each page to explain the output of the print item.
- The *summary* prints some summary information at the end of the report.

The enumerator is a compound role with four subroles: the *input data*, the *empty test*, the *accessor*, and the *step*. These subroles can be filled individually, or they can be filled as a unit with an enumeration cliché (such as *chain enumeration* in Figure 5).

The simple report cliché also includes four constraints, specified at the beginning of the cliché definition. The first constraint specifies "report.txt" as the default name for the file containing the report. This name will be used unless the programmer specifies some other name.

The second constraint specifies that the line limit should be 65 minus the number of lines printed by the print item and the number of lines printed by the summary. Because the line-limit role is computed by this constraint, the programmer never has to fill it explicitly; the role will be updated automatically if the print item or summary is changed. (For example, the line limit is computed to be 64 in Figure 5.)

The remaining two constraints provide default formats for printing items in the report and the column headings. If clichés have been defined for how to print a given type of object in a report and the corresponding headings, then the functions `CORRESPONDING_PRINTING` and `CORRESPONDING_HEADINGS` retrieve the appropriate clichés; otherwise, these functions are simple program generators that construct appropriate code on the basis of the definitions of the types of objects involved. As mentioned earlier, one of the deficiencies of KBEmacs is that the collection of functions (such as `SIZE_IN_LINES`, `CORRESPONDING_PRINTING`, and `CORRESPONDING_HEADINGS`) used in constraints is not easy for the programmer to extend.

General issues in the target scenarios

Before presenting the following two target scenarios, it is important to set the scene by discussing what the Apprentice knows. In each case, the Apprentice is assumed to have an intermediate level of prior knowledge.

If it were possible for the Apprentice to know everything about what the user is going to do, it would be preferable to have a program generator that would construct the desired programs completely automatically. This kind of complete knowledge is possible only in very restricted applications. The Apprentice is intended for use in a broad range of programming applications.

Alternatively, if the Apprentice knew much less than what is illustrated in the scenarios, the user would have to say too much. In particular, the user would either have to define the missing clichés or make do without them. In either case, the productivity and reliability benefits of the Apprentice would be significantly eroded. A major goal of this research, therefore, is to make sure the Apprentice knows enough to be useful.

A second general issue in the two target scenarios is the user interface. Since the interface has not been fully designed, the interactions in the scenarios are intended to illustrate its major features, without being overly specific about details.

For example, input typed by the user is shown in simple English. The Apprentice will not, however, support arbitrary English input. The most important feature of the interface language shown in the scenarios is not the degree of syntactic flexibility, but the use of a large vocabulary of clichés. Because the Apprentice is currently targeted at expert programmers (we can also imagine applications of the technology to computer-aided education of novice programmers), we can assume the user is conceptually familiar with the appropriate clichés. We expect that some combination of training, familiarization, synonyms, spelling correction, and browsing the cliché library will alleviate the problem of needing to know the exact names of the clichés.

Finally, note that several errors have been introduced intentionally into what the user says in each scenario. The particular errors chosen may not appear plausible to all readers. However, many errors are made during the programming process—many of which look obvious with hindsight. The errors introduced here were chosen to illustrate the capabilities of the Apprentice to detect and help correct errors in general.

The Design Apprentice

We are currently building a prototype of a part of the Programmer's Apprentice

called the Design Apprentice, which extends the capabilities of KBEmacs into the realm of design. The target scenario in this section represents our goals in this area for the next three years. Improving on KBEmacs, the Design Apprentice will demonstrate the following three major new capabilities of the Programmer's Apprentice:

- a declarative (specification-like) input language,
- detection and explanation of errors made by the programmer, and
- automatic selection of reasonable implementation choices.

Work on the Design Apprentice began a short while ago with a small programming example similar to Figure 5 but emphasizing the use of Cake to detect and explain programmer errors.

The target scenario for the Design Apprentice concerns the detailed design of a device driver program. This type of program was chosen as a domain for two reasons. First, device drivers have significant practical importance. Second, they are a good example of the kind of domain in which the Apprentice approach is most appropriate—namely, a domain encompassing many similar programs that may nevertheless have some unanticipated idiosyncrasies.

Design clichés. The Design Apprentice's knowledge is embodied in its clichés for typical specifications, typical designs, and typical hardware (see Figure 7). Examples of specification clichés in the device driver domain include initializing, reading, writing, opening, and closing a device. Each of these clichés is annotated with information about what roles and constraints are mandatory, likely, or possible.

Examples of design clichés in the device driver domain include *device driver* and its specializations, such as *printer driver* and *interactive display driver*. Device driver is a very abstract cliché, containing information common to all drivers. The printer driver cliché specifies, for example, that printer drivers support only writing operations and that complex output padding is sometimes required after characters that cause large movements of the print head. Examples of low-level algorithm clichés in the device driver domain include *semaphores*, *busy-wait*, and *watermark processing*.

Examples of hardware clichés that the Design Apprentice knows include *serial line unit*, *printer*, and *interactive display*.

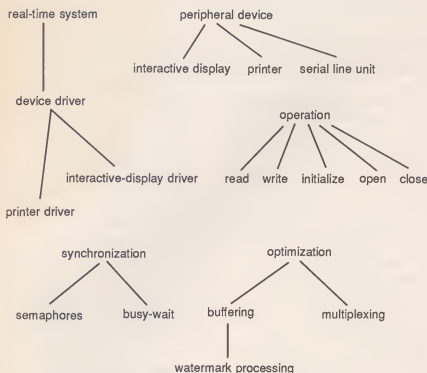


Figure 7. Part of a cliché library for designing device drivers.

device. A serial line unit is a standard bus interface used by many different kinds of hardware devices. It specifies a standard cluster of four buffer and control registers, which are used to operate the device.

Target scenario for program design. Figure 8 shows an interaction between a programmer and the Apprentice at the level of detailed program design. This is the first part of a much longer target scenario⁹ in which this initial program design is further elaborated and changed by the programmer, with assistance from the Apprentice. The interactions in this scenario illustrate the intended use of the facilities in Cake for propagation of information, dependencies, and contradiction handling.

Figure 8 begins with the programmer providing a specification. This could be done interactively, or it could be prepared with a text editor and submitted to the Apprentice all at once. The specification consists of two parts. The first part

describes the hardware (here an imaginary device called the K7). The second part describes a driver program for the K7.

The K7 specification uses the cliché interactive display device. The specification contains both positive information, which describes how particular roles of this cliché are filled in (for example, the screen height), and negative information, which states that some aspects of the cliché are not relevant (for example, the K7 does not support direct cursor positioning).

The last five lines of the K7 specification use the serial line unit (SLU) cliché. The phrase "except that" indicates that the programmer is modifying the SLU cliché rather than just filling in its roles. The exception description uses a number of technical terms ("XCSR," "initialize the device," "blank the screen," etc.) defined for SLUs. This vocabulary enables the programmer to describe the exception succinctly.

The second part of the specification in Figure 8 concerns the driver program. As

with the K7 specification, most of the specification says how various roles of the relevant cliché are filled in (the cliché in this case is interactive display driver).

A particularly interesting part of the driver specification is the implementation guidelines section at the end. The Apprentice uses these guidelines to decide which algorithms to pick when implementing the driver. For example, the first two guidelines cause the Apprentice to select algorithms that have no dynamic storage allocation and that trade time for space. The third guideline instructs the Apprentice to defer inclusion of error-checking code until after the prototype version of the driver is written and tested. The key benefit of this postponement is not that it saves the Apprentice coding time but that it saves the programmer thinking time.

The remainder of the interaction in Figure 8 illustrates the Design Apprentice's ability to detect and explain programmer errors. These errors can be divided roughly into errors of omission (incompleteness) and errors of commission (inconsistency).

Incompleteness can be of two kinds. First, the specification may be missing some expected information which, if provided, would allow the Apprentice to finish the implementation. In this case the Apprentice requests the needed information (for example, the interaction concerning :CLEAR in Figure 8) and proceeds. Second, the Apprentice may simply not have enough knowledge to implement a given specification. In this case the programmer is asked to provide specific implementation instructions (for example, the interaction concerning :LINE-NUMBER in Figure 8). Note that the programmer may postpone answering such a question and is thus able to control the interaction.

Inconsistency can be of two kinds also. There may be inconsistency between different things the programmer says explicitly, and there may be inconsistency between what the programmer says and the knowledge contained in the clichés invoked (for example, the last interaction in Figure 8).

After the problems with :CLEAR and :LINE-NUMBER have been resolved, the Apprentice generates executable code for the K7 driver, as requested. To do so, the Apprentice must make a number of low-level implementation decisions automatically. For example, the Apprentice chooses watermark processing to increase the throughput of the small output buffers. It also makes a number of

reasonable decisions regarding implementation of the various queues and semaphores in the driver. There is not space to show the generated code here, but it is available elsewhere.⁹

The Requirements Apprentice

This section describes a prototype of a part of the Programmer's Apprentice called the Requirements Apprentice, which we are currently building to support the earliest part of the programming process—requirements acquisition and analysis. The target scenario here is excerpted from a longer scenario¹⁰ representing our goals in this area for a three-year effort that began approximately one year ago. Presently, about one third of the longer target scenario is working, and we have formalized approximately half of the requirements clichés involved. Like the Design Apprentice, this prototype uses Cake as the underlying knowledge representation and reasoning system.

Research on requirements acquisition is important for two reasons. From the artificial intelligence perspective, it is a good domain in which to pursue fundamental issues in knowledge acquisition. From the software engineering perspective, studies have indicated that errors in requirements are more costly than any other kind of error. Furthermore, requirements acquisition is not well supported by current software tools.

From informal to formal descriptions.

It is useful to distinguish several phases in requirements acquisition. The earliest phase usually takes the form of a "skull session" whose goal is to achieve consensus among a group of end users about what they want. The requirements analyst's main role in this phase involves his interpersonal skills. The end product is typically an *informal* description of the requirements.

Informal descriptions are characterized by, among other things, incompleteness, ambiguity, contradiction, mis-ordering of information, and the use of undefined terms. These characteristics are not (usually) due to laziness or incompetence. Informality is an essential characteristic of the human thought process. It is part of a powerful debugging strategy for dealing with complexity, which shows up in many problem-solving domains: Start with an

- > The "K7" is an interactive display device where:

The screen height is 4 lines.
The screen width is 40 characters.
The character mode is ASCII.
Direct cursor positioning is not supported.
The keyboard has three keys.

key	character
ACKNOWLEDGE	ACK #0006
YES	Y #0131
NO	N #0116

The bus interface is a standard SLU except that:

Writing a 1 in Bit 1 of the XCSR initializes the device.

Initializing the device blanks the screen and homes up the cursor.
Completion of initialization is signaled the same as transmission of a character.

Sending characters to the K7 and initializing the K7 cannot be done at the same time.

- > The K7 driver is an interactive display driver where:
Echoing is not supported.
Rubout handling is not supported.

The basic software functions are:

Supported: PUTC, WRITE & GETC.

Ignored: OPEN & CLOSE.

Not supported: READ & SEEK.

The control codes are:

:CLEAR - blanks the screen.

:RESET - does a :CLEAR and flushes pending input.

:LINE-NUMBER - returns the cursor line.

The implementation guidelines are:

- 1- Do no conning.
- 2- Favor space efficiency.
- 3- Defer error checking.

- > Write code for the K7 driver.

Request for Advice: I do not know how to support :CLEAR.

Note:

(1) :CLEAR blanks the screen.

(2) The only K7 operation that blanks the screen is initialization.

Unfortunately:

(3) Initializing the K7 homes up the cursor.

(4) A device should not be initialized except by the INIT function.

- > :CLEAR should home up the cursor.

- > Support :CLEAR by initializing the device.

Request for Advice: I do not know how to support :LINE-NUMBER.

Note:

(1) :LINE-NUMBER returns the cursor line.

(2) No K7 operation returns the cursor line.

- > Postpone.

Constraint Violation: There should be a lookahead operation.

Note: The following standard operations provide lookahead:

(1) :ARE-CHARS-WAITING returns T if there are pending input characters.

(2) :NUMBER-CHARS-WAITING returns the number of pending input characters.

(3) :PEEK returns NIL if there are no pending input characters and returns the next character (without removing it from the input buffer) if a character is pending.

- > Use (2).

Figure 8. Hypothetical dialogue between a programmer and the Design Apprentice. Programmer input follows the prompt; response from Apprentice is in italics.

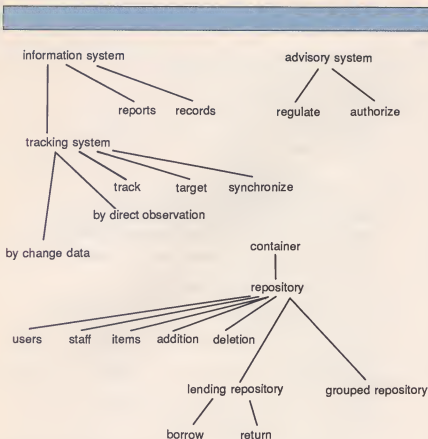


Figure 9. Part of a cliché library for software requirements.

almost-right description and then modify it incrementally until it is acceptable. Thus, dealing with informality in software requirements is not just a matter of being user-friendly—it is a fundamental requisite.

Most current work on software requirements tools focuses on validating *formal* descriptions (descriptions that obey a strict set of mathematical rules of form and content). The main goal of formal validation is to increase confidence that a given formal description actually corresponds to the user's desires by applying simulation, symbolic execution, and various kinds of mathematical analysis. This does not, however, address the key question of how an informal description becomes a formal description in the first place.

The Requirements Apprentice focuses on the transition from informal to formal descriptions. For example, one of our research goals is to elaborate the initial characterization of informality given above, and to develop strategies and

heuristics for removing these features from informal descriptions.

Requirements clichés. As part of our research on the Requirements Apprentice, we are codifying clichés in the area of software requirements (see Figure 9). Compared with implementation and design clichés, the range of clichés involved in software requirements is much more open-ended. In principle, any part of the real world may be relevant to specifying a requirement. In a given application the Apprentice will be useful to the extent that the relevant clichés have been codified.

The target scenario for the Requirements Apprentice concerns the requirements for a hypothetical library system (see Wing¹¹ for a comparison with other approaches to the same example). Three examples of clichés in this area are *repository*, *information system*, and *tracking system*.

A repository is an entity in the physical world. Its basic function is to ensure that

items entering the repository will be available for later removal. A variety of physical constraints apply to repositories. For example, since each item has a physical existence, it can be in only one place at a time and therefore must either be in the repository or not.

There are several kinds of repositories. Simple repositories merely take in items and then give them out. A more complex kind of repository supports the lending of items, which are expected to be returned. Another dimension of variation concerns the items themselves. The items may be unrelated, or they may be grouped into classes. Example repositories include a storage warehouse (simple repository for unrelated items), a grocery store (simple repository for items grouped in classes), and a rental-car agency (lending repository for items grouped in classes).

In contrast to the repository cliché, the information system cliché describes a class of software systems rather than a class of physical objects. The intent of the information system cliché is to capture the commonality between programs such as personnel systems, bibliographic databases, and inventory control systems. Roles of an information system include an *information schema*; a set of *transactions* that can create, modify, or delete the data; a set of *reports* that display parts of the data; *integrity constraints* on the data; a *staff* that manages the information system; and *users* who utilize the information system.

A tracking system is a specialized kind of information system that keeps track of the state of a physical object (the target). The target object is assumed to have a (possibly complex) state and to be subject to various physical operations that can modify this state. The information in the tracking system describes the state of the target object. The transactions modify this information to reflect changes in the target's state.

There are several kinds of tracking systems. A tracking system may

- follow several targets instead of just one,
- keep a history of past states of the target,
- operate on the basis of direct observations of the state of the target or on the basis of observations of operations on the target, and
- participate in controlling the operations on the target, rather than merely observing them.

Example tracking systems include aircraft

tracking systems (tracking multiple targets on the basis of direct observations of their position) and inventory control systems (tracking a repository on the basis of observations of operations that modify its contents).

Target scenario for requirements. The scenario in Figure 10 shows an analyst interacting with the Requirements Apprentice. We are not trying to build a system to be used directly by the end user; we are relying on the interpersonal skills of the analyst to facilitate knowledge acquisition.

The first four commands in Figure 10 begin the process of building a requirement by introducing the new terms *library* and *book*. On the basis of these four commands and the contents of the tracking system and repository clichés, the Apprentice augments its internal model of the evolving requirement in a number of ways.

For example, since the state of a repository is the collection of items it contains (in this case, books), the Apprentice uses the constraints in the tracking system cliché to derive an information schema that provides fields for the three properties listed for books. Also on the basis of the constraints in the tracking system cliché, an expectation is created within the Apprentice that a set of transactions will be defined that correspond to the typical operations on a repository.

Note that the new terms are far from fully defined at this point. They are incomplete because many roles remain to be filled in. They are ambiguous because it is not yet clear which kind of tracking system is intended, or which kind of repository a library is. Since incompleteness and ambiguity are inevitable during the early stages of constructing a requirement, the Apprentice refrains from complaining at this point. It accepts information and performs inferences on a "catch-as-catch-can" basis. However, if requested, the Apprentice can provide a list of currently unresolved issues (see Figure 11) to guide the analyst in finishing the requirement.

The next four commands in Figure 10 begin the definition of the transactions to be supported by the database. The key terms in these commands are *addition* and *removal*: These are two of the standard operations supported by a repository.

The first command to trigger a response from the Apprentice is the analyst's attempt to define the *return* transaction. The Apprentice complains that returning a book, as currently defined, is identical to

- > Begin a requirement for a system called "LIBDB".
- > LIBDB is a tracking system which tracks a "library".
- > A library is a repository for "books".
- > A book has the properties:
 - title - a text string,
 - author - a person's name,
 - ISBN - a unique alphanumeric key.
- > Display the transactions subsection.
- > The "acquire" transaction tracks the addition of a book.
- > The "checkout" transaction tracks the removal of a book.
- > The "return" transaction is the inverse of checkout.
- Problem: The return transaction is identical to the acquire transaction.*
- > Explain.
 - (1) *The return transaction is identical to the acquire transaction.*
 - Because:*
 - (2) *The acquire transaction tracks the addition of a book.*
 - (3) *The return transaction tracks the addition of a book.*
 - Because:*
 - (4) *The checkout transaction tracks the removal of a book.*
 - (5) *The return transaction is the inverse of the checkout transaction.*
- > Retract (4).
- > The checkout transaction tracks the borrowing of a book.
- Note: Assuming a library is a lending repository.*
- > The "retire" transaction tracks the removal of a book.
- > The "purge" transaction tracks the removal of every copy of a book.
- Note: The term every suggests that book is a class. If this is not the case, then the retire transaction is identical to the purge transaction.*
- > Redefine "book" to be a class.
- Book is a class with properties:*
 - title - a text string,
 - author - a person's name,
 - ISBN - a unique alphanumeric key.
- > A "copy" is an instance of a book.
- > A copy has the property:
 - copy number - a number unique within the class.
- > Redefine transactions on a book to apply to a copy.
- Redefining acquire, retire, checkout, and return.*

Figure 10. Hypothetical dialogue between an analyst and the Requirements Apprentice. Programmer input follows the prompt; response from Apprentice is in italics.

acquiring a book. (The equality reasoning facilities in Cake can make this deduction.) This contradicts the Apprentice's expectation that new terms will not be synonymous with old terms unless explicitly declared to be so. Instead of dealing with this problem immediately, the analyst could go on to other things, leaving the problem pending. In this case, however, the analyst decides to deal with the problem right away.

Studying the explanation generated by the Apprentice as requested (using the dependency facilities of Cake), the analyst realizes an error was made earlier in the definition of checkout. The error is cor-

rected by redefining the checkout transaction in terms of borrowing.

The remaining portion of the scenario illustrates the Apprentice's detection of what turns out to be fundamental epistemological confusion on the part of the analyst—between book as a class and book as instances. The analyst decides that title, author, and ISBN are better thought of as properties of a class of books, with each copy as an instance. This conceptual reorganization is propagated by the Apprentice throughout the requirement.

The essential product of interaction with the Requirements Apprentice is its internal representation of the requirements as

3.1.1.1 Checkout

The "checkout" transaction tracks the borrowing of a copy of a book from the library.

INPUTS: ISBN number and copy number.

OUTPUTS: none.

PRECONDITIONS: The copy of the book uniquely identified by the inputs must be in the roster of copies of books which are in the library.

EFFECT ON THE INFORMATION STORE: The input is borrowed from the roster of copies of books which are in the library.

UNUSUAL EVENTS: If the input is not in the roster of copies of books which are in the library, then the information system is inconsistent with the state of the repository. A notation is made in the error log.

USAGE RESTRICTIONS: none.

Unresolved issues:

Should historical record keeping be added?

Should checking of user validity be added?

Should checking of staff member validity be added?

Figure 11. Portion of a hypothetical requirements document to be produced by the Requirements Apprentice at the end of the dialogue in Figure 10.

frames and plans in Cake. This representation will eventually feed into the Design Apprentice. In the meantime, however, this representation can be used to answer queries and to generate various documents for the requirements analyst, the end user, and the system designer.

Figure 11 shows the kind of document the Apprentice will be able to generate. As with natural language input to the Apprentice, the syntactic aspects of good English are not the key concern here. What is important is deciding *what* to say by, for example, choosing the appropriate level of detail.

The goal of the prototypes described above is to demonstrate the feasibility of the cliché-based assistant approach to automatic programming using the technology of the Plan Calculus and Cake. We plan eventually to connect the Design Apprentice and Requirements Apprentice as a first demonstration of a complete Apprentice.

In addition to these prototypes, the Programmer's Apprentice project also includes a number of other investigations based on the same approach and underlying technology. For example, an intelligent debugging assistant¹² has been built that uses the Plan Calculus and the dependency-directed reasoning capabilities of Cake to help a programmer localize bugs. A reverse engineering system⁵ has been demonstrated that, given a program and a library of clichés, applies

graph-parsing techniques to the Plan Calculus to construct a plausible design of the program in terms of the clichés. An experiment in program translation has been performed¹³ using reverse engineering and the Plan Calculus. A new programming-language construct¹⁴ has been developed embodying the clichéd forms of iteration. Finally, the propositional, algebraic, and frame layers of Cake have been packaged into a separate utility, which could be used to build assistants in other engineering domains.

The Programmer's Apprentice is a basic research project. However, some of the principles and technology described here are beginning to move into practical application. Frame-based knowledge representations and dependency-directed reasoning are beginning to appear in the repertoire of techniques for computer-aided software engineering. For example, Bachman Information Systems, with our consultation, has developed and is marketing an intelligent assistant for database design based in part on the Programmer's Apprentice. Sanders Associates, also with consultation from the Programmer's Apprentice project, has developed a knowledge-based requirements assistant (see the article by Harris summarized in the sidebar covering the companion issues of *IEEE Expert* and *IEEE Software*). We hope these efforts are just the leading edge of the transfer of this research into commercial practice. □

The Programmer's Apprentice project is a collaboration involving students, faculty, and staff over several years. In connection with the work described in this article, we would particularly like to acknowledge Yishai Feldman, who has been responsible for the development of Cake for the past several years; Howard Reubenstein, who is working on the Requirements Assistant; and Yang Meng Tan, who is working on the Design Apprentice.

This article describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research has been provided by several organizations, including the National Science Foundation under grant IRI-8616644, IBM Corp., NYNEX Corp., Siemens Corp., and the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K-0124.

The views and conclusions expressed in this article are those of the authors and should not be interpreted as representing the policies, expressed or implied, of the supporting organizations.

References

1. C. Rich and R.C. Waters, "Automatic Programming: Myths and Prospects," *Computer*, Vol. 21, No. 8, Aug. 1988, pp. 40-51.
2. E. Soloway and K. Ehrlich, "Empirical Studies of Programming Knowledge," *IEEE Trans. Software Eng.*, Vol. 10, No. 5, Sept. 1984, pp. 595-609.
3. K.E. Huff and V.R. Lesser, "A Plan-Based Intelligent Assistant That Supports the Process of Programming," *Proc. Third Symp. Software Development Environments*, ACM, New York, Nov. 1988.
4. C. Rich, "A Formal Representation for Plans in the Programmer's Apprentice," *Proc. Seventh Int'l Joint Conf. Artificial Intelligence*, Morgan Kaufmann, Los Altos, Calif., 1981, pp. 1,044-1,052.
5. L.M. Wills, "Automated Program Recognition," Tech. Report 904 (master's thesis), Artificial Intelligence Lab, Massachusetts Inst. of Technology, Sept. 1986.
6. C. Rich, "The Layered Architecture of a System for Reasoning About Programs," *Proc. Ninth Int'l Joint Conf. Artificial Intelligence*, Morgan Kaufmann, Los Altos, Calif., 1985, pp. 540-546.
7. R.C. Waters, "The Programmer's Apprentice: A Session with KBEmacs," *IEEE Trans. Software Eng.*, Vol. 11, No. 11, Nov. 1985, pp. 1,296-1,320.
8. C. Green and D.R. Barstow, "On Program Synthesis Knowledge," *Artificial Intelligence*, Vol. 10, No. 3, Nov. 1978, pp. 241-279.
9. C. Rich and R.C. Waters, "The Programmer's Apprentice: A Program Design Scenario," Memo 933A, Artificial Intelligence Lab, Massachusetts Inst. of Technology, Nov. 1987.

10. C. Rich, R.C. Waters, and H.B. Reubenstein, "Toward a Requirements Apprenticeship," *Proc. Fourth Int'l Workshop Software Specifications and Design*, IEEE-CS Press, Los Alamitos, Calif., Apr. 1987, pp. 79-86.
11. J.M. Wing, "A Study of 12 Specifications of the Library Problem," *IEEE Software*, Vol. 5, No. 4, July 1988, pp. 66-76.
12. R.I. Kuper, "Automated Techniques for the Localization of Software Bugs," Tech. Report 1053 (master's thesis), Artificial Intelligence Lab, Massachusetts Inst. of Technology, May 1988.
13. R.C. Waters, "Program Translation Via Abstraction and Reimplementation," *IEEE Trans. Software Eng.*, Vol. 14, No. 8, Aug. 1988, pp. 1,207-1,228.
14. R.C. Waters, "Using Obviously Synchronizable Series Expressions Instead of Loops," *Proc. 1988 IEEE Int'l Conf. Computer Languages*, IEEE-CS Press, Los Alamitos, Calif., Oct. 1988.

(Editor's note: References 2, 4, 7, and 8 are reprinted in *Readings in Artificial Intelligence and Software Engineering*, C. Rich and R.C. Waters, eds., Morgan Kaufmann, Los Altos, Calif., 1986.)



Charles Rich is a principal research scientist at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology, where he has worked since 1980. He and coauthor Waters are the principal investigators of the Programmer's Apprentice project. Rich's research interests are knowledge representation and the application of artificial intelligence to engineering problem solving, especially in software engineering.

Rich received a bachelor's degree in engineering science from the University of Toronto and master's and doctor's degrees in artificial intelligence from MIT. He is a member of the Assoc. for Computing Machinery and the American Assoc. for Artificial Intelligence as well as the IEEE Computer Society.



Richard C. Waters is a principal research scientist at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology, where he has worked since 1978. He and coauthor Rich are the principal investigators of the Programmer's Apprentice project. Waters' other interests include programming languages and engineering problem solving.

Waters received a bachelor's degree magna cum laude in applied mathematics (computer science) from Brown University, a master's degree in computer science from Harvard University, and a doctor's degree in artificial intelligence, with a minor in linguistics, from MIT. He is a member of the Assoc. for Computing Machinery and the American Assoc. for Artificial Intelligence and a senior member of the IEEE Computer Society.

Readers may write to the authors at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139.

SEND FOR THE REST OF THE STORY

A FREE trial copy of *IEEE Expert*

Our sister magazine, *IEEE Expert* regularly addresses artificial intelligence and expert system issues. Articles in the November issue were specifically chosen to supplement the information in this issue of *Computer*.

And you can get a **FREE TRIAL COPY** if we hear from you by December 31, 1988.

Simply fill out and return the attached subscription card, and we will enter your subscription to *IEEE Expert* and send your free trial issue.

If you like it, simply pay the invoice when you receive it. If you don't, just write "Cancel" on the invoice and return it. The November issue is yours to keep in either case.

A limited number of magazines have been printed for this offer. Send in your card today!

A FREE trial copy of *IEEE Software*

Our sister magazine, *IEEE Software* regularly addresses software engineering issues. Articles in the November issue were specifically chosen to supplement the information in this issue of *Computer*.

And you can get a **FREE TRIAL COPY** if we hear from you by December 31, 1988.

Simply fill out and return the attached subscription card, and we will enter your subscription to *IEEE Software* and send your free trial issue.

If you like it, simply pay the invoice when you receive it. If you don't, just write "Cancel" on the invoice and return it. The November issue is yours to keep in either case.

A limited number of magazines have been printed for this offer. Send in your card today!

Reducing Contention in Shared-Memory Multiprocessors

Per Stenström
Lund University

A multiprocessor with shared memory contains a number of processors that share part of the available memory. The processors can execute distinct programs as well as cooperate in the same application by using the shared memory for process communication. This makes them suitable for a broad range of applications.

The past few years have seen an increasing interest in shared-memory multiprocessors. Several new multiprocessors have been announced every year. One of the main problems with this type of computer, however, is how to avoid performance degradation due to the memory sharing. Although there are other important design issues, this article concentrates on the techniques we can use to design a memory system that reduces the impact of contention. To exemplify the techniques, I will review 10 implementations and the design decisions taken in each.

To demonstrate the problem of sharing, Figure 1 shows a simple example of a processor-memory organization for a multiprocessor. The processors share a number of memory modules. An *interconnection network* establishes the sharing.

Since the processor-memory traffic is proportional to the number of processors, the bandwidth of the interconnection network must be proportional to the number of processors to avoid performance degradation. For multiprocessors containing a large number of processors, the resulting

Contention for shared resources such as memory and an interconnection network limit the performance of shared-memory multiprocessors. The implementations reviewed here demonstrate ways to reduce contention.

interconnection network can become complex and expensive. Thus, it is important to reduce the processor-memory traffic as much as possible.

Even if the interconnection network meets the processor-memory bandwidth requirement, performance degradation can result if several processors contend for a common link in the interconnection net-

work. This type of contention is referred to as *communication contention*.

Another kind of contention relates to the fact that a memory module can handle only one memory request at a time. When several processors request the same memory module, the requests are serialized. The contention that then arises is referred to as *memory contention*.

In the next section, we will investigate some standard techniques to design a memory system for a shared-memory multiprocessor that reduces communication and memory contention.

Techniques for reducing contention

The organization of the memory and the design of the interconnection network are two keys to the reduction of contention. Therefore, we will first investigate some different memory organizations and their impact on contention. Subsequently, we will investigate the design principles of interconnection networks.

Memory organization. Main memory is partitioned into a number of memory modules. A memory module can be *local*, meaning that it attaches to one processor, or *global*, meaning that it is only accessible by sending requests through the interconnection network. A request from a

processor to its local memory does not cause any network traffic. A multiprocessor with local and global memory modules appears in Figure 2.

Local memory modules can be divided further into *shared* and *private* memory modules. Shared memory modules are accessible by all processors, whereas a private memory module is accessible only by the processor to which it is attached. Global memory modules are implicitly shared, and private memory modules are implicitly local. Consequently, there are three types of memory modules: local/private; local/shared; and global/shared.

Private memory provides a means for reducing network traffic and, consequently, memory and communication contention. Allocating private data structures to private memory means that requests for such data structures do not cause network traffic.

Private memory has the disadvantage that it cannot be used for shared data structures. Another approach uses local/shared memory modules, which can be used for both shared and private data.

The number of shared memory modules has a great impact on memory contention. If the number of shared memory modules is less than the number of processors, memory contention will occur if all processors issue a shared memory request at the same time.

Interconnection network. The main task of the interconnection network is to establish the sharing of memory modules between the processors. We will investigate the most important interconnection networks and their main characteristics.

The interconnection network must meet the memory bandwidth requirement of the processors. For a multiprocessor to be *modular*, meaning that the bandwidth requirement is fulfilled when adding processors, the bandwidth must be proportional to the number of processors.

Another parameter is the *common access throughput*, meaning the maximum number of simultaneous requests that can pass through the interconnection network at the same time. For multiprocessors with as many global memory modules as the number of processors, we would like the common access throughput to equal the number of processors, to reduce communication contention.

The *latency time*, meaning the time it takes for a request to pass through the interconnection network, is another important parameter. Because the inter-

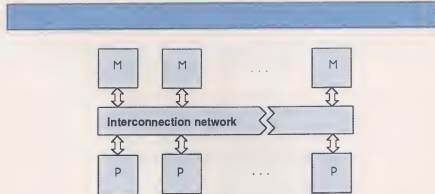


Figure 1. An example of a processor-memory organization for a shared-memory multiprocessor.

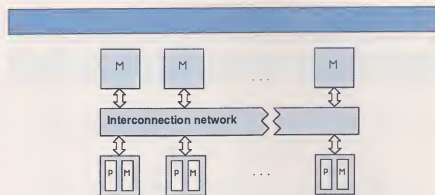


Figure 2. A multiprocessor with local and global memory modules.

connection network is part of the critical path between a processor and a global memory module or a local/shared memory module in another processor, the latency time adds to the total memory access time. We therefore want to reduce the latency time.

Common bus. The most attractive kind of interconnection network—due to its simplicity—is the *common bus*. However, a common bus has a fixed bandwidth and, consequently, is not modular. The common access throughput for a bus is one; it can handle only one request at a time.

A bus arbiter selects one of several requesting processors to get exclusive access to the common bus. An obvious way to increase the bandwidth and the common access throughput is to use several parallel, independent buses capa-

ble of handling several memory requests at the same time.

Typically, the bus latency time (or bus cycle time) is shorter than the memory access time. To take full advantage of the bandwidth of the bus, designers often implement a bus protocol that allows pipelining of several memory requests. While one memory module executes a load, another processor can send a memory request to another memory module.

Crossbar switch. We can view a *crossbar switch* as a number of vertical and horizontal links interconnected by a switch in each intersection (see Figure 3). The number of vertical and horizontal links equals the number of processors and memory modules, respectively.

The crossbar switch is the ultimate solution for high performance. First, it is a

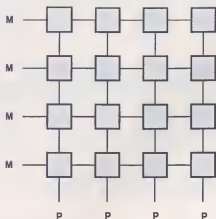


Figure 3. A 4x4 crossbar switch.

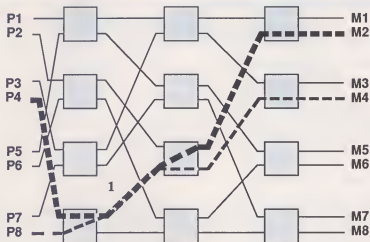


Figure 4. An example of an 8x8 multistage network. This type of multistage network, called an Omega network, is characterized by a perfect shuffle interconnection preceding every stage of switching elements.

modular interconnection network in that the bandwidth is proportional to the number of processors. Second, the common access throughput is the same as the number of processors. Communication contention can never occur, since contention for a link can only arise when more than one processor requests the same memory module. However, the crossbar has the

disadvantage that the number of switches is N^2 , given N processors and memory modules.

Multistage networks. An interconnection network of less complexity than the crossbar, the *multistage network*, contains a number of switching elements resembling crossbar switches. The switching ele-

ments typically have the same number of inputs and outputs. Assuming k inputs and outputs (referred to as $k \times k$ switching elements), the switching elements are organized in $\log_k N$ stages with N/k switching elements in each stage, given N processors and memory modules as shown in Figure 4. They interconnect in such a way that one unique path exists between every processor-memory pair. A request must pass through all stages of switching elements to reach its final destination. The latency time thus equals $O(\log N)$.

Routing of memory requests typically occurs in a distributed fashion in the following way, assuming 2×2 switching elements. Suppose that the destination address is $\langle d_0 d_1 \dots d_{m-1} \rangle$, where $N = 2^m$. If the outputs of a switching element are numbered 0 and 1, then the switching element in stage i sends the message to output d_i . The distributed routing scheme permits pipelining of memory requests, which increases the throughput of the network.

Multistage networks have a common access throughput that equals the number of processors. The peak bandwidth is $O(N)$ and the number of switching elements is $O(N \log N)$, which reduces the complexity to a crossbar switch for multiprocessors with a large number of processors.

A complication of multistage networks is that contention for a link can arise even when requests are routed to different memory modules. This occurs because some of the links in one processor-memory path are sometimes common with links in another processor-memory path, which can result in communication contention. In Figure 4, two processors (4 and 8) send memory requests to different memory modules (2 and 4). However, both requests must pass along link 1.

One way of resolving contention for a link is to buffer requests at the switching elements. When a memory request is blocked, it is queued in the switching element until the link is free. Another possibility is to simply remove the request from the network and let the issuing processor retransmit it after a while.

We can provide an alternative path for each processor-memory pair by adding an extra stage of switching elements, used when contention for the primary path arises. This approach also improves the fault tolerance property of the network.

We can reduce the latency time and the number of conflicts by using switching elements with a larger number of inputs.

However, this increases the complexity and cost for the resulting network.

Several processors might repeatedly access the same memory module, as happens when several processors try to acquire a lock. The kind of contention that then arises is usually called *hot-spot contention*. It causes memory contention because all memory requests to the "hot" memory module will be serialized. In multistage networks with buffered switching elements, studies have shown that the contention is propagated back to the issuing processors in a tree-like fashion. Since each link is common to several processor-memory paths, this can also result in communication contention for other memory requests than those for the "hot" memory module.

Designers have employed the technique of *combining* to reduce performance degradation caused by hot-spot contention. Combining works in the following way. If a switching element observes two memory requests to the same memory location, say two reads, then it sends only one request to the next stage and queues the other. Because combined requests can be further combined, only one memory request will reach the memory module in the ideal case, independent of the number of processors that issue a read.

Numerous papers have been published on various aspects of interconnection networks for multiprocessors. The interested reader can consult the special issue of *Computer*, Vol. 20, No. 6, June 1987, on interconnection networks and the collection of papers in *Interconnection Networks for Parallel and Distributed Processing*.¹

Memory allocation. Even if the interconnection network meets the bandwidth and common access throughput requirements of the processor-memory traffic, memory contention can still cause a problem if the processor-memory traffic concentrates on a small number of memory modules. Therefore, it is essential to consider how data structures are allocated to the shared memory modules.

Local memory modules (shared or private) can store code and data structures private to the processors to which they attach, resulting in reduced communication and memory contention.

For shared data structures, the allocation method becomes crucial to memory contention. We can reduce memory contention for shared data structures by having several memory modules and an

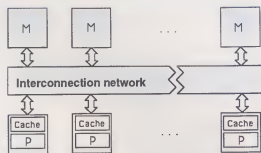


Figure 5. Private cache organization for a multiprocessor.

interconnection network with a common access throughput that equals the number of shared memory modules. Distributing the shared data structure uniformly across the shared memory modules permits distributing memory requests over the memory modules, thus reducing memory contention. Software can support the distribution of shared data structures; the compiler allocates the data structure so that memory references are scattered uniformly across the memory modules. Alternatively, hardware can interpret parts of the address as the memory-module identifier (a technique called *interleaving*).

Cache memory. In uniprocessors, a cache reduces the effective memory access time. In multiprocessors, a cache can also reduce network traffic (and, consequently, communication contention) as well as memory contention. To do this, we can attach a private cache to each processor, as shown in Figure 5. This cache organization reduces memory traffic because the cache can satisfy most memory references.

One of the problems with private cache organizations—the *cache consistency* or *cache coherence* problem²—is introduced by the existence of several copies of a shared memory block. This can introduce inconsistency if special arrangements are not provided to detect when one copy is modified. Note that inconsistency can occur only for shared, writable memory blocks. Read-only or nonshared data structures can always be safely cached without precautions.

Let's discuss some standard techniques for solving the cache consistency problem.

One solution to the cache consistency

problem is to cache only data structures that cannot cause inconsistency. One obvious example is read-only data structures such as code. An improvement to this method is to also cache shared, writable data structures during periods when they are modified by only one processor. This can be done under software control; the compiler tags data as *cacheable* or *non-cacheable*. Locks are examples of non-cacheable data.

We can regard data in critical sections as cacheable because only one processor can modify the critical data at a time. However, the modification of the critical data must be reflected in main memory when the process exits the critical section. We can do this by employing *write-through*, where all writes update main memory, or *copy-back*, where the block is written back when the process exits the critical section. In either case, the cached copy of the block has to be invalidated to prevent referencing of stale data. The performance of the software tagging scheme relies on the ability of the compiler to detect the cacheability of data structures.

Another solution maintains consistency of shared data by specific cache consistency protocols implemented in hardware. In bus-oriented multiprocessors, consistency is maintained by a bus-watching mechanism in each cache, called the *snoopy cache controller*, which implements a cache consistency protocol.

Archibald and Baer surveyed and evaluated several of the proposed cache consistency protocols.² In the simplest one, called *write-through with invalidation*, all write operations are broadcast to the other caches. If a copy of the modified block

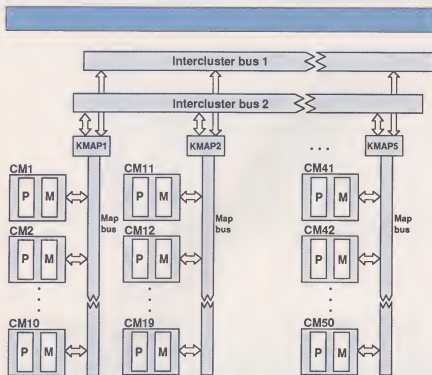


Figure 6. System architecture of CM*.

exists in a cache, then this copy is invalidated. A more efficient variation of this protocol, referred to as *write-through with update*, updates all copies of a shared block instead of invalidating them. A problem with these protocols is that all write operations cause network traffic.

A protocol that circumvents part of this problem is the one referred to as *write-ownce*. The first write invalidates all other copies of the block. However, unlike the write-through-with-invalidation protocol, subsequent writes to the same block can be done locally because the cache knows that it has the only valid copy. If another cache requests a copy of that block, then either memory or another cache responds to the cache load request, depending on which of them has the most up-to-date copy of the requested block.

Other protocols are based on ownership of a block. We refer to this class of protocols as *ownership protocols*. If it is known a priori that a block is going to be modified, then all copies of that block can be invalidated when the block is loaded into the cache by a special ownership acquisition command. The invalidation on the first write operation to that block can

then be eliminated.

The performance of these protocols relies on the ability to specify whether the block is going to be modified prior to a cache load operation. With assistance from the compiler, such actions can be specified.

For other interconnection structures such as crossbar switches, broadcast operations are more costly than in bus-oriented architectures. In a proposed scheme, a bit-map is associated with each memory block administered by the memory controller. The consistency traffic is reduced by letting the bit-map reflect which caches have a copy of the memory block. Cache commands (invalidations or updates) can then be sent exclusively to the caches that have a copy of the memory block, minimizing network traffic.

Another disadvantage with private caches is that the load traffic can become prohibitive if several processors need to access the same memory block. In this case, all caches must load that memory block. Letting a number of processors share one cache reduces the load traffic, but then we must consider contention for the shared cache.

Synchronization and contention. Synchronization is a fundamental concept in parallel programming because it provides the basis for cooperation of tasks in a program. Several synchronization schemes have been proposed, all based on some fundamental synchronization primitive carried out as an *atomic operation*, that is, without interruption.

One example is the Test&Set operation with the following semantics:

```
function Test&Set( lock );
begin
    temp := lock;
    lock := 1;
    return( temp );
end;
```

The old value of "lock" is read and "lock" is then set to 1 indivisibly. To acquire the lock, the processor must wait until the lock is released. This is typically done by *spin-lock*, letting the processor repeatedly execute the Test&Set operation.

One implementation of Test&Set in a multiprocessor environment works in the following way. The processor reads the lock and writes back a 1 to the memory location. To guarantee that no other processor is modifying the lock variable, this operation is done indivisibly by memory interlock.

Obviously, this operation could be more efficiently implemented by letting the memory controller execute it. In such an implementation, a Test&Set generates no more network traffic than does one memory load, thus reducing the traffic caused by spinning. However, the processor still has to repeatedly execute Test&Set, which can lead to contention.

If we implement a hardware cache consistency protocol, we can reduce contention by caching the lock variable. If we do not allow the synchronization variable to be cached, then the resulting traffic caused by spin-locks can become prohibitive. If we use a multistage network, one possible technique to reduce network traffic for spin-locks is combining (discussed earlier).

To demonstrate the techniques employed in multiprocessor implementations, let's investigate the design decisions taken in 10 implementations. Some of the implementations are commercial systems and others are experimental prototypes. I have included the experimental prototypes to demonstrate some novel ideas on how to reduce memory and communication contention.

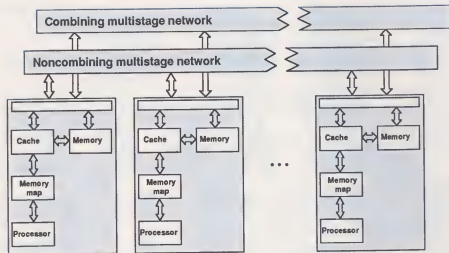


Figure 7. System architecture of RP3.

Some multiprocessor implementations

C.mmp. One of the earliest multiprocessors implemented is C.mmp at the Computer Science Department at Carnegie Mellon University.³ It contains 16 slightly modified PDP-11 computers and 16 global memory modules interconnected by a crossbar switch. In addition, each PDP-11 contains local/private memory.

I think it worth mentioning that the designers planned for each PDP-11 to contain a cache to reduce memory contention. Cache consistency would then be maintained by caching only read-only or non-shared data such as code and process stacks.

Communication contention is eliminated by the use of a crossbar, and memory contention is reduced by the private caches and local/private memory of the processors.

CM*. A subsequent design at Carnegie Mellon, CM*,³ was designed as an experimental base for research projects focusing on architectures for loosely coupled, message-based multiprocessors; tightly coupled, shared-memory multiprocessors; operating systems, and algorithm design.

Figure 6 shows the organization of CM*. It contains 50 computer modules, called CMs. Each CM contains an LSI-11

computer and a local/shared memory module in addition to the private memory associated with the LSI-11.

A hierarchical interconnection structure reduces communication contention. The CMs are partitioned into five clusters, with CMs in a cluster interconnected by a common bus called a Map bus. Clusters interconnect through a packet-switched bus called an Intercluster bus. To increase the bandwidth for intercluster communication, the designers used two Intercluster buses.

The local/shared memory organization permits memory references in different CMs to take place at the same time, which reduces communication contention. A second order of improvement is accomplished by the cluster structure, which permits memory references in different clusters to take place at the same time. The cost for this, however, is nonuniform memory access times to different parts of the shared memory.

The Kmaps shown in Figure 6 are microcoded controllers used to support different communication mechanisms and synchronization schemes.

RP3. Several research prototypes containing more than a hundred processors are currently being built at different sites. Researchers building one of them, RP3,⁴ at IBM's T.J. Watson Research Center in Yorktown Heights, New York, are invest-

igating both hardware and software aspects of highly parallel computation.

RP3 will contain 512 processing elements. The organization of each PE appears in Figure 7. Each PE contains a processor, local/shared memory, and a cache.

Two multistage networks interconnect the PEs. The first multistage network has the ability to combine memory requests directed to the same memory location. The second network is a low-latency, noncombining multistage network.

Routing control is distributed to the switching elements in both networks. Memory requests are pipelined, and contention for links is resolved by queueing requests in each switching element. The noncombining network is built from 4×4 switching elements, and the combining network is built from 2×2 switching elements.

From the programmer's point of view, local memory is partitioned into private and shared parts. Code and shared data structures are typically allocated to the shared part of local memory and uniformly scattered across the shared memory modules to reduce memory contention. The memory address mapping hardware does this. The least-significant physical address bits are interpreted as the PE number, which results in consecutive addresses being located in different

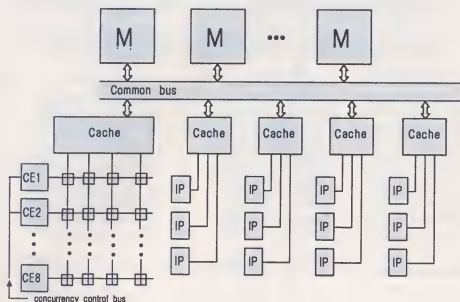


Figure 8. System architecture of Alliant FX/8.

memory modules (a technique called *fine interleaving*).

A cache in each PE reduces network traffic and the effect of network latency. A software tagging scheme maintains cache consistency. Different logical entities of data can be invalidated, such as segments, pages, and blocks.

A class of atomic operations referred to as Fetch&Op supports synchronization. One example, Fetch&Add, works in the following way: Fetch&Add(V, B) returns the old value of V and adds B to the old value of V indivisibly. Fetch&Add can increment the loop index when different processes execute loop iterations. The operation (in this case, Add) is executed in memory by specific hardware support mechanisms in the memory controller.

Since shared variables such as locks are noncacheable, repeated execution of Test&Set, for example, leads to hot-spot contention. Combining reduces hot-spot contention. Each switching element in the combining multistage network can combine two Fetch&Ops. The latency time for the combining network is longer due to the switch complexity caused by combining. For this reason, other memory requests than those subject to hot-spot contention

are routed through the noncombining low-latency network.

Alliant. Alliant Computer Systems' Alliant FX⁵ is a commercial multiprocessor with vector processing capability, intended for computation-intensive applications. The organization of the Alliant FX/8 appears in Figure 8.

The Alliant FX/8 contains eight computational elements (vector processors) and 12 interactive processors used for general-purpose applications. The CEs form a *computational complex*. All processors share a global memory. A common bus connects the global memory modules to the processors. To use the bus bandwidth efficiently, several memory requests can be in progress at the same time; while one memory module executes a load, the bus can serve another request.

Five private caches reduce bus traffic. All CEs share a four-way interleaved cache, connected to the CEs by an 8×4 crossbar switch. Three IPs share a cache (see Figure 8). A write-once cache consistency protocol maintains cache consistency. The IPs contain local/private memory that stores frequently used kernel code.

Alliant specifically designed the computational complex to combine vector and parallel processing. Iterations within a Do-loop, for example, can be executed in parallel by the computational elements. Data dependencies can sometimes prevent one computational element from continuing. The concurrency control bus that interconnects all CEs facilitates efficient synchronization by sending interprocessor messages.

Cedar. Another multiprocessor built as part of a research project is Cedar,⁶ a project at the Center for Supercomputing Research and Development at the University of Illinois, Urbana-Champaign. Compiler and operating system design are among the topics studied within the project.

Cedar contains eight Alliant FX/8 computational complexes, called clusters, interconnected with 64 global memory modules by a multistage network (see Figure 9). The Alliant computational complexes are slightly modified to contain a port to the multistage network. The multistage network, built from 8×8 switching elements, contains two stages.

The memory system consists of a hier-

archy of local/private memory within a cluster and global memory shared between the clusters. The researchers reduced memory and communication contention by efficiently exploiting the hierarchical memory organization. The local cluster memory acts as a cache for the global memory. Software controls and maintains cache consistency. Nonshared data structures can be cached without inconsistency problems. Allocating tasks in a clever way keeps most memory references local within a cluster.

Tasks that show a fine granularity of parallelism can be efficiently allocated to the same cluster. The Alliant concurrency control bus provides a means for synchronizing these tasks. A global synchronization scheme supports synchronization between tasks in different clusters. The synchronization processors attached to each global memory module support a variety of synchronization primitives such as Test&Set.

Butterfly. BBN Laboratories' Butterfly⁷ is a commercial, general-purpose multiprocessor with shared memory. The multiprocessor, designed to be modular, ranges from a configuration of a few processors to a full configuration of 256 processors. It consists of a number of processor nodes interconnected by a multistage network built from 4×4 switching elements (see Figure 10a). Routing is distributed as for multistage networks (discussed earlier). The network has an alternative path for each processor-memory path, used when contention for a link arises.

Each processor node consists of a processor, local/shared memory, and an interface to the interconnection network, as shown in Figure 10b. To reduce memory and communication contention, data structures are allocated in the following way: Code copies exist in each processor node. The process stack and other private data structures are allocated locally. Shared data structures, however, are scattered uniformly across the memory modules by the memory allocator in the compiler to reduce memory contention. The processor node controller implements synchronization primitives such as Test&Set.

SPUR. Multiprocessors also suit workstations. A university project along this line, called SPUR, is being conducted at the University of California at Berkeley.

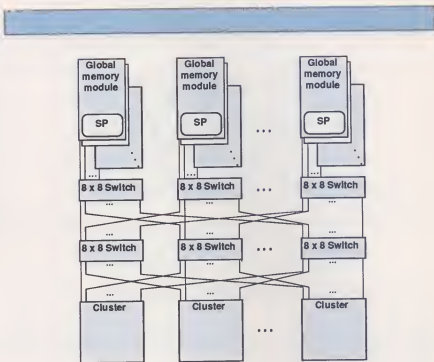


Figure 9. System architecture of Cedar. SP stands for synchronization processor.

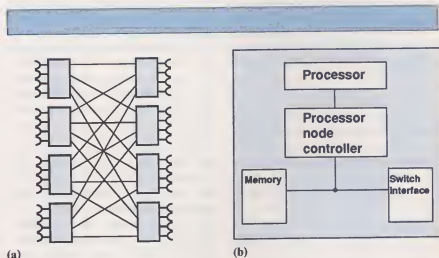


Figure 10. Interconnection network of a 16-processor Butterfly configuration (a) and processor node organization (b).

SPUR⁸ contains up to 12 processors. Each processor board contains a processor, a floating-point unit, and a cache, as shown in Figure 11. A common bus (the SPUR bus) connects the processor boards to the global memory modules.

A large private cache is attached to each processor. The cache consistency problem is solved by a protocol referred to as the *Berkeley ownership protocol*. It works basically the same way as write-once. Unlike write-once, however, ownership

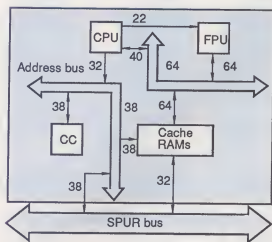


Figure 11. Organization of a processor board in SPUR. (Source: *Computer*, Vol. 19, No. 11, Nov. 1986, p. 10.)

can be acquired by special bus transactions causing all copies to be invalidated. When a block is owned, all write operations can be carried out locally. The owner is responsible for updating main memory when ownership changes.

Dragon. Dragon is another workstation, designed for internal use at Xerox Palo Alto Research Center.⁹ It contains up to 10 processors interconnected with the global shared memory modules by a common bus.

A cache is attached to each processor. Cache consistency is maintained by employing a write-through protocol for shared blocks. All write operations are broadcast to the other caches through the common bus. The snoopy mechanism in each cache checks whether its cache contains a copy of the block and updates it correspondingly. For nonshared blocks, write operations are carried out locally in the cache (copy-back memory-update policy), which means that bus traffic is not generated for nonshared data structures.

Multimax. Some commercial multiprocessors combine a number of off-the-shelf microprocessors to achieve a high performance/cost ratio. Encore Computer's Multimax¹⁰ contains 11 processing

elements and a number of global memory modules, interconnected by a high-speed, pipelined bus.

To take full advantage of the bus bandwidth, the bus is not held while waiting for memory. Instead, several memory requests can be pipelined. Memory contention is reduced by interleaving memory requests across several global memory modules.

Each processing element contains two processors and a cache. Letting two processors share a cache reduces the load traffic caused by code and other frequently accessed shared data structures.

Using write-through with invalidation maintains cache consistency. Snoopy mechanisms in each cache monitor every bus access, supporting invalidation of every copy of a block when the block is updated.

The hardware support for synchronization is an indivisible Test&Set instruction. The memory controller executes Test&Set. The cache reduces bus traffic caused by spin-locks.

Balance. Another similar design is Sequent Computer Systems' Balance.¹¹ The Balance 21000 consists of 15 processing elements interconnected with global memory modules by a pipelined, common

bus. The bus protocol implemented permits several memory requests to be pipelined so as to efficiently exploit the full bandwidth of the bus.

Each processing element contains two processors, a cache, and local/private memory. Cache consistency is maintained in the same way as in Multimax. To further reduce bus traffic, the local/private memory stores some frequently used kernel code. A dedicated lock memory connected to the bus supports process synchronization primitives. The lock memory supports Test&Set directly in hardware. Cache memory reduces bus traffic when a processor is spinning on a lock.

Comparison

Different design decisions have been made to reduce memory and communication contention, depending on the size of the multiprocessor in terms of the number of processors. Typically, multiprocessors containing a small number of processors (such as Alliant, SPUR, Dragon, Multimax, and Balance) use a global memory organization in conjunction with a common bus. Bus traffic in these systems is mainly reduced by large, private caches. A common bus supports efficient implementation of hardware-implemented cache consistency protocols.

The systems reviewed here demonstrate several protocols. Multimax and Balance use write-through with invalidation. This protocol generates bus traffic for all write operations and works efficiently only if the fraction of write operations is small. The Dragon multiprocessor exemplifies an improvement to this protocol, with write operations to nonshared blocks performed locally in the cache. Another improvement is that write operations to shared blocks update all other copies instead of invalidating them.

An ownership protocol is applied to SPUR. A mix of software and hardware support controls the actions of the protocol efficiently.

Pipelining has been used in the bus-oriented multiprocessors to take full advantage of the bandwidth of the bus. It permits the bus to be used while a processor waits for a response from memory.

CM* uses a quite different bus organization. The hierarchical bus system and the local/shared memory organization reduce communication contention.

A crossbar switch proves suitable when high bandwidth is required. C.mmp,

Table 1. Main characteristics of the multiprocessors reviewed.

System	Number of Processors	Shared Memory	Interconnection Network	Cache Memory	Cache Consistency
C.mmp	16	Global	Crossbar switch	Yes	Software tagging
CM*	50	Local	Hierarchical	No	—
RP3	512	Local	Multistage network	Yes	Software tagging
Alliant	20	Global	Common bus	Yes	Write-once
Cedar	64	Hierarchical	Multistage network	Yes	Software tagging
Butterfly	256	Local	Multistage network	No	—
SPUR	12	Global	Common bus	Yes	Ownership
Dragon	10	Global	Common bus	Yes	Write-through with update
Multimax	22	Global	Common bus	Yes	Write-through with invalidation
Balance 21000	30	Global	Common bus	Yes	Write-through with invalidation

which contains 16 processors, uses a crossbar switch. The Alliant FX/8 uses a crossbar switch to establish a high-bandwidth interconnection between the eight computational elements and the shared cache system.

The limited bandwidth of a common bus is not sufficient for multiprocessors containing a large number of processors. A crossbar switch is not an alternative, either, because of the complexity of a large number of processors. RP3, Cedar, and Butterfly use a multistage network. In RP3 and Butterfly, the multistage network contains several stages. Contention in these networks is resolved differently; switching elements are buffered in RP3, while an alternative processor-memory path is provided in Butterfly. The multistage network in Cedar consists of large switching elements (8×8), which restricts the number of stages to two. Contention, therefore, is not an important issue in Cedar.

In RP3, private caches reduce network traffic and memory access time. Software maintains cache consistency in this case. In Butterfly, which does not contain private caches, code and private data structures are allocated to the local memory of each processor module. To reduce memory contention for shared data structures, these systems use the same number of memory modules as processors. Furthermore, shared data structures are scattered across the memory modules to distribute memory requests evenly. In RP3, data structures and code are distributed by means of hardware interleaving. In Butterfly, the compiler allocates the shared data structure so as to uniformly distribute memory references over the memory modules.

Cedar's designers took a different approach, applying a hierarchical memory

organization. The global/shared memory serves as a backing store for the local/private cluster memory, which acts as a cache. The designers of both RP3 and Cedar had to consider the cache consistency problem. Software controls and maintains cache consistency in these systems.

Synchronization in most systems is based on the Test&Set operation. The memory controller executes Test&Set to reduce network traffic. An important issue is to reduce memory and communication contention caused by spin-locks. In the bus-oriented systems such as Multimax and Sequent, caching the lock variable reduces contention. The cache consistency protocol ensures the atomicity of the synchronization primitive.

In the systems that make use of a multistage network, spin-locks might cause memory and communication contention. In RP3, combining requests reduces contention caused by several processors accessing the same location in memory.

Table 1 shows the main characteristics of the multiprocessors reviewed here. The number of processors is the maximum number supported for the systems. Shared memory is classified in accordance with the type of memory modules: global means that only global memory modules are used, local means that only local/shared memory modules are used, and so forth.

Experimental evaluation of design principles

We have investigated several techniques to reduce network traffic and communication and memory contention. We can conclude that bus-oriented multiprocessors

suit only multiprocessors containing a small number of processors. The maximum number of processors depends on several parameters, such as processor memory reference rate, cache size and organization, cache consistency protocol, bus bandwidth, and memory access time.

For multiprocessors that contain a large number of processors, multistage networks seem the only possible alternative today. Multistage networks, though less complex than crossbar switches, are expensive and tend to dominate the price of a multiprocessor. Although a common bus has a fixed bandwidth, its advantages make it an interesting subject for investigations into methods to reduce bus traffic. Such is the objective of an ongoing project in the Department of Computer Engineering at Lund University in Sweden.

To reduce memory and communication contention, researchers in the project have made the initial design decision to distribute the shared memory with local/shared memory modules in each processing element.

Memory allocation principles provide some of the keys for reducing contention in multiprocessors with shared memory. By gaining more knowledge of memory sharing in parallel applications, designers can implement communication mechanisms based on VLSI technology that reduce bus traffic. One obvious example is cache consistency protocols that have been shown to reduce network traffic considerably.

One of the studies in the project evaluates virtual memory management schemes for multiprocessors with local/shared memory. Some of the systems reviewed here support virtual memory. In these systems, address translation is based on page

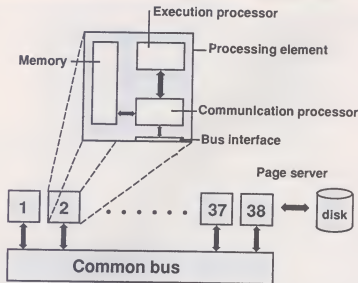


Figure 12. System architecture for the multiprocessor emulator.

tables stored in memory. To reduce the address translation time, translation buffers (special-purpose caches for page table entries) are attached to each processor. This also reduces network traffic, but introduces consistency problems for the same reason as for private caches.

In a bus-oriented multiprocessor with local/shared memory, designers can reduce network traffic by distributing virtual memory management to the processing elements. In the scheme proposed, each local/shared memory module keeps track of all its pages by storing the page table entries in an accompanying fast memory. This scheme has the advantage that it entirely eliminates bus traffic for local accesses. In addition, it avoids consistency problems because there are no copies of page table entries.

Another study concerns the performance of cache consistency protocols. The implementations reviewed here demonstrate a large number of cache consistency protocols. The performance gained depends on the program behavior of the parallel applications.

To study principles for memory and communication structures based on VLSI in a multiprocessor with local/shared memory, researchers

in the Lund University project have designed a multiprocessor emulator.¹² The multiprocessor emulator consists of 38 processing elements interconnected by a common bus as shown in Figure 12.

The processing elements use microprocessors to emulate parts of the hardware mechanisms with software in the following way: Each processing element contains two processors (see Figure 12). The first one—the execution processor—executes the application program. The second one—the communication processor—emulates memory and communication mechanisms. Every memory reference issued by the execution processor is handled by software executing on the communication processor. The communication processor uses the local/private memory in each processing element to emulate memory structures. The common bus provides interprocessor communication between the communication processors at different processing elements.

The simplicity of emulating a wide range of hardware mechanisms constitutes the main advantage of our emulator. We can take measurements by including monitoring mechanisms in the emulated hardware, which is implemented in software running on the communication processor.

On the other hand, the emulator has the

disadvantage of limited performance. However, a performance evaluation has shown that each processing element can execute 1,000 emulated memory references per second, which is sufficient for experimental purposes.

Our emulator is suitable for evaluating the performance of different design decisions under real workloads. To provide real workloads for the experiments, we have implemented a parallel Ada system on the multiprocessor emulator. Choosing Ada makes it possible to use existing parallel Ada programs in the experiments, thus eliminating the work of writing realistic applications ourselves. The Ada system supports dynamic allocation and reallocation of tasks and memory at runtime, thus enhancing the experimental capabilities of the multiprocessor emulator.

Shared-memory multiprocessors have succeeded mainly because they provide a high performance-price ratio for a broad range of applications. Processor speed continues to increase, which will stress the contention aspects further in coming years. For shared-memory multiprocessors to be competitive, the contention problem must be solved in an economical way.

I think that there is still too little understanding of parallel program behavior. When parallel programming has become a part of the programmer's daily life, deeper knowledge about the interaction of parallel activities will result naturally. The designer can use this knowledge for new solutions to the contention problem.

New solutions might include, for example, allocation principles for shared data structures, or how to implement cache consistency protocols that take into account behavioral aspects of parallel programs. VLSI technology makes it feasible to implement sophisticated protocols of this kind, which I think will play an important role. □

Acknowledgments

I'm indebted to Lars Philipson for all his valuable discussions and continuing support of my work. I would like to thank all of the referees for the many suggestions that improved this article.

This work was supported by the Swedish National Board for Technical Development (STU) under contract numbers 80-3962, 83-3647, and 85-3899.

References

1. C.-L. Wu and T.-Y. Feng, eds., *Interconnection Networks for Parallel and Distributed Processing*, IEEE Computer Society Press, 1984.
2. J. Archibald and J.-L. Baer, "Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model," *ACM Trans. Computer Systems*, Vol. 4, No. 4, Nov. 1986, pp. 273-298.
3. K. Hwang and F.A. Briggs, *Computer Architecture and Parallel Processing*, McGraw-Hill, Hightstown, N.J., 1984.
4. G.F. Pfister et al., "The IBM Research Parallel Processor Prototype (RP3): Introduction and Architecture," in *Proc. 1985 Int'l Conf. Parallel Processing*, Oct. 1985, pp. 764-771.
5. Alliant, *FX/Series Product Summary*, tech. report, Alliant Computer Systems Corp., Littleton, Mass., Oct. 1986.
6. D.J. Kuck et al., "Parallel Supercomputing Today and the Cedar Approach," *Science*, Vol. 231, Feb. 28, 1986, pp. 967-974.
7. BBN, *Butterfly Parallel Processor Overview*, tech. report, BBN Laboratories Inc., Cambridge, Mass., Mar. 1986.
8. M. Hill et al., "Design Decisions in SPUR," *Computer*, Vol. 19, No. 11, Nov. 1986, pp. 8-22.
9. L. Monier and P. Sindhu, "The Architecture of the Dragon," in *Proc. 30th IEEE Computer Society Int'l Conf.*, 1985, pp. 118-121.
10. Encore, *Multimax Technical Summary*, tech. report, Encore Computer Corp., Mar. 1987.
11. Sequent, *Balance Technical Summary*, tech. report, Sequent Computer Systems Inc., Nov. 1986.
12. P. Stenström and L. Philipson, "A Layered Emulator for Design Evaluation of MIMD Multiprocessors with Shared Memory," in *Proc. PARLE, Parallel Architectures and Languages Europe, Lecture Notes in Computer Science*, Vol. 1, No. 258, Springer-Verlag, Amsterdam, June 1987, pp. 329-344.



Per Stenström is an assistant professor of computer engineering at Lund University in Sweden. His research interests include parallel architectures, VLSI, and performance evaluation for parallel systems.

Stenström spent the academic year 1987-1988 in the Computer Science Department at Carnegie Mellon University as a visiting scientist. There he participated in the PIE project (Programming and Instrumentation Environment for parallel processing).

He received an MS degree in electrical engineering in 1981 and a Licentiate of Technology degree in computer engineering in 1987, both from Lund University.

Readers may write to the author at Lund University, Dept. of Computer Engineering, PO Box 118, S-221 00 Lund, Sweden.



KING FAHD UNIVERSITY OF PETROLEUM & MINERALS Dhahran - Saudi Arabia

College of Computer Science and Engineering COMPUTER ENGINEERING DEPARTMENT

The Department of Computer Engineering seeks applications for faculty positions at all levels. Rank and salary commensurate with qualifications. Applicants must hold a Ph.D. degree in Computer Engineering or related areas. Individuals with research interests in one or more of the following areas are preferred: Digital System Design, Computer Architecture, Computer Networks, Data Communications, VLSI, Design Automation, Fault Tolerant Computing, and Microprocessors & Computer Based Systems. Department labs include Design Automation, Digital Design, Microprocessor Systems, and Robotics. Teaching and research are supported by a VAX-11/780, fully equipped Graphic Center, and a number of Microcomputers as well as a University DPC that includes AMDHAL 470V6 and IBM 3090 mainframes.

The University offers attractive salary and benefits plus air conditioned furnished housing and air transportation to and from Dhahran each year. Ten months duty each year with two months vacation salary. Minimum regular contract for two years, renewable.

Apply before December 31, 1988, with complete resumé and supporting documents to:

Dean of Faculty & Personnel Affairs
KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DEPT. NO. COE/382
DHAHRAN — 31261, SAUDI ARABIA

Using Inspections to Investigate Program Correctness

Robert N. Britcher
IBM Corporation

Inspections are central to contemporary software standards and development planning. More incisive than a review, they make the private practice of programming public. Programmers can no longer carry on their business with the computer in secrecy; they must hold it up to criticism and account for their errors. Peers observe their work and them. Inspection reports are generated. Follow-up work is required, and sometimes a follow-up inspection. Statistics are kept. Software quality is evaluated based partly on the number of errors discovered in inspections.

Inspections can apply to any software document: requirements, specifications, formal design, code, test plans, etc. They are administered by project. Often, an independent and strategic organization—quality assurance—approves the project's procedures for conducting inspections, monitors their conduct, and collects, synthesizes, and reports the resulting metrics.

Inspections have not changed significantly since their introduction. (In 1976, Fagan published a comprehensive approach to inspections as developed and used at IBM.¹) A typical code inspection begins three or four days before the inspection meeting. Invitations issued to the prospective participants describe where and when the meeting will occur, what will be inspected, and who will attend—typically a moderator, the author, a presenter, a representative from quality

An approach to software inspections that examines the properties underlying correct programs would help programmers verify correctness and build it into their programs.

assurance, and some number of inspectors. The invitation includes the material to be inspected (the exhibit). If the exhibit is source code, for example, the material could include the design specification from which the code was derived and the unit test plans and procedures.

The moderator leads the meeting and prepares the inspection report. Participants must report the amount of time spent preparing for the meeting, reading and inspecting the exhibit, and document-

ing their findings. The presenter reads aloud from the exhibit, usually not line-by-line. The prologue, commentary, data specifications, and logic are subject to questions, clarifications, and identification of errors. The errors and questions are recorded by category and referenced to the exhibit. The meeting lasts about two hours.

Afterwards, the moderator produces a summary report, documents in the project database any design issues that may have surfaced, signs the report, and distributes it. The report describes the inspection and its results, including the number of lines of code, commentary, data, etc. inspected; whether another inspection is required; and the number of minor and major errors found in the specification, design, logic or data specification, commentary, interface, standards, and so on.

Inspections offer the first opportunity for a third party to uncover errors not found by the author or the author's tools. More significantly, inspections can reveal errors in intent and in the underlying semantics of the software. The typical error categories suggest that inspections as commonly practiced do not probe deeply enough. A program written according to the standards and compiled without error can still be incorrect. That is, it may not always execute correctly under the conditions and constraints that will befall it throughout its history, particularly if it is written to support a complex, real-time

system. Inspections should focus more on the hidden area behind the recorded material, asking such questions as "Is the software intrinsically reliable?" and "How resilient is it?"

A new approach to inspections—one that would augment current methods and could be used selectively to test its worthiness for the program at hand—would emphasize the search for correctness. It would hold up to scrutiny not only what is on the page, but also the thought behind the representation. The recording of the program would not be an issue; exhibits, although necessarily small, could be at the highest level of development. Inspectors instead would investigate how the program developed, looking for evidence of disciplined methods in its construction, adequate consideration of the error domain, and the program's ability to withstand years of use and inevitable change.

The example inspection in this article uses a program developed according to IBM's design methods. The inspection presents the arguments as a series of questions that the inspectors would ask the author and themselves. (Parnas² recently explored the practice of using questionnaires at inspections with good results.) I selected this simple example to illustrate only the principles described in this article, not to illustrate the size and complexity of a typical program covered in a two-hour inspection meeting.

Developing correct programs

The search for correctness in an inspection presupposes a method for writing correct programs. For example, IBM has adopted a standard program-design method that presumes programs have a mathematical quality and can be constructed algebraically.

Mills likens an algebra of proper (single-entry, single-exit) programs built on the simplicity of three prime-control structures (sequence, conditional statement, and loop) to the composition of integers built on the operators +, -, *, and /. Thus, the integer 14 decomposes into $2 + (4 * 3)$; the operators + and * are introduced "independent of any other terms in the arithmetic expression."³

As programs expand, formal verification techniques check their correctness; $2 + (4 * 3)$ is verified as equivalent to 14. Programmers successively invent new configurations of operators, having deter-

mined the correctness of their predecessors. This notion applies equally to operands; programmers can invent abstract data and expand its meaning. The objective is to reduce a complex algorithm to its simplest form, specifying it in terms of abstract operators and data, then decomposing it step by step into programs of less abstract operators and data that the computer can ultimately execute.

We can model the abstract operators and data as functions or as state machines. A function represents the mapping of inputs to outputs, or $o = f(i)$. A state machine⁴ is a function with internal states that we can represent as the set of ordered pairs $\{(i, \text{old}), (o, \text{new})\}$, where old is the state prior to execution and new is the state after execution. When enacted as programs, the function becomes the variables and instructions that will produce o , given i (a procedure). The state machine becomes the collection of procedures that might modify the state. For example, a state machine implemented as an Ada package would specify the inputs, the outputs, and a private set of state data accessible to users of the package only by invoking one of the procedures.

IBM developed an Ada process-design language to record program designs. (The example in this article does not use Ada/PDL, but a PDL similar to Pascal developed in the 1970s.) Using the PDL, a programmer models a program as a function or as a state machine, specifying its intended behavior (for example, $a := \min(x, y)$ assigns the minimum value to a). He then refines the specification (if $x < y$ then $a := x$ else $a := y$) and verifies the refined program's correctness with respect to its specification. Shankar⁵ asserts that "by modeling the program as a function, one can give a much more uniform, definite, and clearer meaning to the concept of stepwise refinement. Program correctness can be defined very precisely by giving a set of proof rules for each step."

Investigating correctness

Thousands of programmers have learned the principles of state machines, algebraic decomposition, verification, and so on. However, because these principles tell programmers how to reason about programs, measuring results is difficult. When reading a program at an inspection, how does one know that it has been decomposed and verified incrementally?

Further, how does one explain the utility of these principles? Does algebraic decomposition really improve the chances of correct execution?

A programmer needs to apply only the few principles described above to write a correct program. The principles are turned upside down, though, when we read a program to ascertain its correctness. We must discover the effects of these principles by advancing the right questions or arguments to obtain some measure of correctness. Inspections often find only the obvious defects in programs. Can they find defects in reasoning, given a baseline for developing correct programs like the one described above?

I leave the question of using formal proofs in inspections to the participants. Both positive⁶ and negative conclusions regarding formal proofs have appeared in the literature. (Although I do recommend rigorous methods for investigating correctness, the techniques are not those defined by IBM⁷ for formally verifying control structures and modules as part of its approach to developing correct programs through stepwise refinement.)

Correctness arguments

The example in the next section uses correctness arguments based on four key program attributes: *topology*, *algebra*, *invariance*, and *robustness*.

Topology here refers to the relationship of a program to the problem space (for example, an algorithm to track aircraft) over the range of its developmental life. For example, a 100,000-line program is not developed as one program but as a succession of programs descendant from a single program or specification. Would that program be more reliable if it evolved through a dozen (stepwise) refinements, or even as few as three? I use algebraic correctness as Mills defines it: each intermediate program should be functionally equivalent to its antecedent. Invariance explores the relationship between variables. Robustness investigates how thoroughly the programmer considered the error domain.

Topology. Structured programming, which uses algebraic replacement to develop proper programs, is an accepted practice that is generally believed to yield more reliable programs than unstructured programming. This reliability stems from the structured program's mathematical

lineage—each program builds on its predecessor, which is a simpler (and, intellectually, more manageable) version of its successor. The final program is the culmination of many historical programs, each of which solved a relatively small problem, enclosing a small problem space. The structured program's topology is rooted in its history: the small size of the problem space is conserved as the program expands. In turn, the program's reliability is rooted in its topology: the smaller the problem space, the more likely it becomes that a solution is correct.

Structured programming helps programmers arrange their thinking so that they can create a procedure of great simplicity. Data abstraction (modeled here as a state machine) encourages them to create the simplest pattern among many procedures by discovering the simplest pattern in the data space. The state machine enables programmers to make sense of large programs, to create abstract data and the abstract operators that empower it. The state machine conserves the problem space of data with respect to operators, both historically (as the data and its operators are refined) and structurally (as the final programs). The ultimate configuration of programs designed as state machines emerges as a set of small, independent processing specifications, like the abstract data types described by Hoare⁷ and Guttag.⁸

The question in this portion of the inspection, then, is whether the author conserved the problem space in developing the operators and data, resulting in small, independent programs solving small, independent problems.

Algebra. Few large-scale programming problems present themselves as a well-defined collection of small problems such that 50 programmers can set off to write several hundred small programs. Rather, these problems are discovered through a succession of alternately divergent and convergent processes in which analysts and programmers find a software system's requirements, produce a design, learn more about the system, change the design accordingly, and so on. A software system of one million lines of code can take years to code, not because it takes years to code 5,000 programs, but because it takes years to define them. In the extreme, developing large software systems amounts to reducing the unknown to a precise and correct set of executable statements through a succession of intermediate steps, each

A large chunk of software in real-time systems accounts (or should account) for errors.

demanding the invention of new abstractions. Correctness is most at risk in this middle territory.

Correct programs must be correct in terms of their syntax and semantics. Their structure and developmental history should reflect the designer's concern for and attention to correctness. The objective of the algebraic argument is to discover how well the program adheres to the author's (or his or her predecessor's) vision of it. In short, is the statement "if $x < y$ then $a := x$ else $a := y$ " equivalent to its forebearer, the more abstract and less procedural specification, " $a := \min(x, y)$ "?

Invariance. Executing a program generates a sequence of states, or values. Contemporary programming languages enable us to give attributes to states because the computer and its microcode contain embedded types, or processing specifications, such as integer and character. We can allocate types to memory locations, give them identifiers (names), and refer to them as variables. Variables give the programmer a window to the computer's states and allow him or her to perceive it as a mechanical object capable of doing work, rather than as a mathematical object. The epigram "state machine" indicates that it is both.

Procedural programming languages now encourage the specification and creation of data types⁸ and variables. There is general agreement that the "data is as important as the logic." Although the idea of treating the computer as a mathematical object is starting to make an impact, the end game in programming remains writing instructions. Investigating and absorbing the relationships among variables before using them in a program requires considerable patience. (Shaw's article on abstraction in programming is quite readable.⁹) Gries¹⁰ refers to programming as a "goal-oriented activity,"

where the goal is maintenance of the correct and intended relationship among variables before, during, and after the execution of an algorithm. The programmer should "know the properties of all objects to be manipulated by the program" because "the more properties you know" the more likely it becomes that you can "create an (elegant) algorithm."

The invariance argument will shape the inspection most. It dictates that the inspectors and the author investigate the properties of variables and the relationships among them—the program's states—formulating, for example, the program's postcondition and precondition, loop invariants, and terminating conditions. This suggests a more active inspection than currently practiced, with inspectors and the author describing their thoughts and positions on paper.

Robustness. Because errors and changes in intent arise during the development process, software products and systems only attain their final and fullest form when they are coded and tested. In the early phases of development, programmers and analysts try to discover and document the work they will automate. Whether written in a natural language or graphically, whether formal or informal, the functional specifications describe the designer's vision of the system. They become the best information available about the software's purported behavior.

At their best, specifications describe what the system will do when it performs correctly. However, they seldom describe what the system will do when things go wrong. Most often, that problem is left for the programmers to work out in a succession of design and code decisions. The subtleties of the system's behavior are introduced in its design, coding, and testing, particularly in large, complex systems. One of the ironies of the software life cycle is that the specifications, as much as they influence the implementation, are themselves influenced by the implementation, sometimes profoundly.

A large chunk of software in real-time systems accounts (or should account) for errors, including those introduced by the user, the user manuals, the computer, the programmer, and neighboring systems. Errors are intrinsic to artificial systems, just as they are to biological systems. They may even enrich a system. Programs embedded in software systems or large software products not only must execute correctly with respect to their specifica-

tions, but also must support the environment in which they will operate. That support primarily involves dealing with errors: understanding their causes, analyzing them, reporting them, and minimizing their effects on the user. Embedded programs often assume the best or nothing at all, ignoring the error domain.

Because so much depends on the individual programmer, the inspection should not ignore the relationship between the program and its operating environment. The robustness argument views the program as an organ of the system (or product) in which it is embedded. Inspectors should determine the system's robustness and how well it tolerates errors. Put differently, they should determine if the precondition of a program is weak enough to accommodate all the inputs imported to it, and if the postcondition is strong enough to ensure that all its outputs are within the range expected by its user.

Applying correctness arguments

To inspections using correctness arguments, the author would invite but two or three inspectors whose only necessary qualification is an interest in the inspection—because their program uses the subject program, because they are interested in software correctness, because the subject program represents the evolution of their design, etc.

The material would be available to the participants prior to the meeting, allowing enough time to read and evaluate it with respect to the correctness arguments. The meeting would be led, although whether by the author or an inspector is not important. The author should not defend the exhibit. The meeting would last no more than two hours, and the leader would record the evaluation and the inspectors' recommendations for improving or correcting it.

The exhibit. The inspection exhibit could be any piece of design or code recorded in a procedural language. Since the decisive issue is correctness, the exhibit need not be executable. It should be compilable (and correctly compiled) to set the inspectors at ease and demonstrate the author's willingness to have the exhibit inspected for his or her approach rather than for agility with the language. Using the same lexicon for design and coding

The robustness argument views the program as an organ of the system in which it is embedded.

would simplify the investigation of a program's algebraic decomposition. (Ada has an advantage here, since it can serve as a design language and a coding language.)

The specification from which the subject program derives would comprise an integral part of the exhibit. In the case of the algebraic argument, the aim would be to inspect the program with respect to the specification, not to inspect the specification itself.

The exhibit must be small, perhaps a single state machine. The author should simplify the representation and explicitly define all data types and variables used by the procedure, identifying them with short names.

The example. The program in this article was developed in 1981 from a succession of state machines to implement a message switching network.¹¹ One of the intermediate-level state machines provides secure access to a host message processor from a configuration of intelligent workstations. This workstation manager, resident in the message processor, supervises the active user sessions and processes their keyboard inputs and display outputs.

The subject program described below is one procedure developed from the specification of one of the 20 fully refined state machines that resulted from the workstation manager's decomposition. The state machine is the dispatcher, and its state data consists of

- a list of identifiers naming the workstations;
- a list of identifiers naming the functions (the type of work) assigned to the workstations;
- a list of identifiers naming the processes (the supervisory software's notation for tracking the allocation of the message processor, its channels, and its devices to the system work)

assigned to the functions; and

- a list of identifiers naming the current events (such as "a keyboard-read is pending") associated with each workstation.

The name of a workstation control block is paired with each of the names in these lists. The block contains information about the workstation's internal and external states, including the names of the workstation, function, process, and event to which it is mapped. (The workstation names are not explicitly defined in the state machine.)

The `map-process_identifier` procedure implements one of 13 operators that act on the objects described above. It returns to a requesting program the name of a workstation control block, having received the name of a process.

The exhibit for the example program would include the specification of the dispatcher state machine and the procedure (see Figures 1 and 2). The list, `session_pids`, in the procedure is the list of identifiers naming the processes. The state machine defines this variable, and the procedure acts on it. The procedure appears here as it was found in the library, neither compilable nor executable. The target language is an IBM assembly language.

The mechanics. The author and the inspectors would collaborate in the pursuit of correctness, developing questions and answers centering on topology, algebra, invariance, and robustness, and working through the program's underlying patterns. They would suggest ways to improve the program—including identifying any errors—and evaluate the methods used by the programmer and the project.

Topology. In working through the topological argument, the participants would develop the answers to the following questions:

- Does the program (procedure) represent a function derived from a state machine?
- Is the state data defined precisely?
- How many variables comprise the state data? Are they related? Does the state represent a coherent set of values? (A large and/or heterogeneous collection of variables could represent a state machine that did not conserve a small problem space as it was refined. The resulting module would be difficult to verify and maintain.)

module TR11S580.workstation.dispatcher state machine

```
definitions
  references
    TR11S020.workstation.manager.types

  design
    input
      var vdu_evt : events
      var physid : workstation_ids
      var logid : workstation_functions
      var procid : process_ids
      var vduent : workstation_control_block_name

    output
      var vduentry : workstation_control_block_name
      var result : boolean
      var physvid : workstation_ids
      var logvid : workstation_functions

    state
      var vdu_ids : list of workstation_ids
      var vdu_fun : list of workstation_functions
      var session_pids : list of process_ids
      var vdu_events : list of events

    initial state
      vdu_ids = empty
      vdu_fun = empty
      session_pids = empty
      vdu_events = empty

    procedures
      .
      .
      .
      map-process-identifier (in procid out vduentry)
      "map a process id onto a workstation"
      .
      .
      .

end module
```

Figure 1. Workstation dispatcher state machine.

- How large is the state space? Is the range of values represented by the variables too large?
- Does the program act on an inordinately high percentage of the variables in the state space? (A state machine consisting of only a single procedure, for example, would sug-

gest that the system or product had not been decomposed uniformly and consistently.)

- Does the program modify the state or just refer to it?
- Does the program modify or refer to state data in another state machine? (Such a design would severely limit

implementation options and would create maintenance problems.)

- How many local variables were invented in the procedure and what is their rationale?
- How complex is the procedure? How many predicates does it contain and how deeply are they nested?
- Does the program consist of proper programs that are easy to identify and replace with more-abstract proper programs?

This list is not complete; investigators should formulate new and different questions and rephrase the above questions.

We know that the example program was derived from a state machine and that the state is coherent. The values themselves are identifiers, and each manifests an independent and coherent function. The program operates on only one identifier and does not modify the state. It is a simple program that appears to be the culmination of a topologically sound design.

Algebra. In determining the exhibit's soundness from a historical or developmental standpoint, the investigators must ask whether the exhibit was decomposed and refined algebraically or partitioned in an ad hoc way. Questions would include:

- Is the state machine specified in an abstract and precise way, or does it just repeat the information of an earlier (higher-level) specification?
- Is the program specified precisely, as in the form $o = f(i)$?
- Is the program's input domain and output range a subset of the inputs and outputs defined by the state machine specification?
- Does the exhibit define the initial state?
- Does it precisely define the range of possible values in the state?
- Does it precisely specify the proper programs that compose the program, and does the specification appear in the program?
- Is each proper program functionally equivalent to its specification?

The example program is specified precisely in the form $o = f(i)$. Map-process-identifier maps the input procid onto the output vduentry (see Figure 2). The state machine specification is precise, and the initial state of the lists is defined and set to "empty." The program is simple and reasonably well-specified, although the commentary preceding each proper program is not precise. The state

space and the local variables are not typed explicitly. The participants would not know the possible values for a process identifier. The implied state (null) is vague; is it an identifier, a value, or both? Null does not appear in the state machine specification, so the participants would be left wondering how it would be used in the lists. This is a major shortcoming in the exhibit, for the entire algorithm turns on it.

Invariance. Making the invariance argument would be arduous, even in this small program. The participants would start by simplifying the identifiers: p for `procid`, v for `vdentry`, e for `session_pids`. The variable for integer, i , would remain the same. Here the investigators would discover a marked oversight in the design. They were told that each identifier (workstation, function, process, or event) in each list is paired with an identifier naming the workstation control block. The state machine specification defines the lists as a set of identifiers. The participants would assume that each element in the list is a pair of identifiers: (`process_identifier`, `workstation_control_block_name`).

The participants must define two states invented by the author but not specified: null (n) and the `workstation_control_block_name` (w). They must also assume the worst about n : that it is both an identifier and a value.

To gain more insight into the problem, the participants would record all they know about each variable. For example, i is an integer used to index the list `session_pids`; `procid` is the input argument and an identifier whose value is itself an identifier; and w is an identifier that defines an identifier (the name of a workstation control block) and occurs in every element of every list except the last element, which is undefined but appears to contain the value n .

The participants could now define e more precisely as a list of type $\{e\{p, w\} \text{ and } e\{n, n\}\}$. Further, the expression $e(i)$ —an index of the process list—defines a function: given p , w is returned; given (NOT) p , n is returned.

After defining all the variables in the problem and understanding their roles clearly, the participants would construct a table of the form shown in Table 1, where B describes the variables' behavior (their possible values and relationships to the other variables) before execution of the algorithm, D describes the behavior during execution, and A describes it after execution. Working through the relationships

```
proc map_processor_identifier (in procid out vdentry)
```

```
"search the process id list for a match with the input"
"argument; return the name of the workstation control"
"block associated with that entry; or null, if no match found."
```

```
var procid: process_ids
var vdentry: workstation_control_block_name
var i: integer
```

```
"search list for process id"
i := 0
while
  procid (not) = session_pids(i) &
  session_pids(i) (not) = null
do
  i := i + 1
od

"return the name or null"
case
  session_pids(i)
part (procid)
  return name (i)
part (null)
  return null
esac
```

corp

Figure 2. Map-process_identifier program.

of the variables would assure the team that the author understood the problem in depth and solved it correctly. The table is a platform; in fact, the participants need not construct a formal table. Like the questions, its purpose is to structure the search for correctness.

Robustness. To argue robustness, the participants would ask:

- How rich is the input domain? How many variables are imported and how coherent are they?
- How rich is the output range? Is enough information conveyed to the program user?
- Is the range of values explicit in the input domain and the output range?
- Does the program guard the input domain?
- If the program invokes other functions that are defined in other state machines, are the functions, results, and possible error returns defined explicitly?
- Is the domain of each predicate explicitly defined so that each can be evaluated as true or false?

Table 1. Invariance table for map process identifier.

id	B	D	A
e			
w			
p			
v			
i			
n			

- Does the program define all conditions that could result in an error?
- Is the meaning of each error clear?
- Does this program return error values to its users and are their meanings clear? (They may end up in the hands of a system or product user.)

In the example, the input domain and output range of the program are clearly specified, but the possible values are not defined explicitly. The program does not adequately test the input domain. No externally defined functions are invoked. The predicate domains are explicit. However, the null returned to the user of this program remains ambiguous. The state machine specification sets the initial state of the lists to "empty," but the program does not consider the empty state, unless null is synonymous with empty.

In summary, participants in a formal inspection of this program would have discovered significant problems with the design, although much about it is sound. In the process of investigating the program's history—more to the point, the history of the programmer's thinking—the participants would have learned something about its future, and the programmer would have had the opportunity to change its future and the future of all programs he or she might develop.

As we develop better tools for recording and compiling software designs and code, those

who think about and practice programming will take greater interest in the more obscure aspects of a program: its intent, meaning, resilience, and developmental history. Although the problem of writing correct programs, especially those embedded within large systems or products, remains largely unsolved in practice, the situation is improving. We can use inspections to further the investigation into how correct programs are constructed. Several such inspections will be carried out to determine their usefulness and refine their practice. The purpose of incorporating correctness arguments into inspections is not to improve inspections, but to improve programming. This is not a modest objective. Steps will necessarily be small. □

Acknowledgments

I would like to thank the referees for their helpful reviews and Pat D'Orazio for her ideas and editorial assistance.

References

1. M.E. Fagan, "Design and Code Inspections to Reduce Errors in Program Development," *IBM Systems J.*, Vol. 15, No. 3, 1976, pp. 182-211.

2. D.L. Parnas and D.M. Weiss, "Active Design Reviews: Principles and Practices," *J. Systems and Software*, No. 7, 1987, pp. 259-265.
3. R.C. Linger, H.D. Mills, and B.I. Witt, *Structured Programming Theory and Practice*, Addison-Wesley, Reading, Mass., 1979, p. 150.
4. A.B. Ferrentino and H.D. Mills, "State Machines and Their Semantics in Software Engineering," *Proc. First Int'l Computer Software and Applications Conf. (COMP-SAC)*, CS Press, Los Alamitos, Calif., 1977, pp. 242-251.
5. K.S. Shankar, "A Functional Approach to Module Verification," *IEEE Trans. Software Eng.*, Vol. SE-8, No. 2, Mar. 1982, pp. 147-160.
6. E. Dijkstra, *A Discipline of Programming*, Prentice Hall, Englewood Cliffs, N.J., 1976.
7. C.A.R. Hoare, "An Axiomatic Basis for Computer Programming," *Comm. ACM*, Vol. 12, Oct. 1969, pp. 576-580.
8. J.V. Guttag, "Notes on Type Abstraction (Version 2)," *IEEE Trans. Software Engineering*, Vol. SE-6, No. 1, Jan. 1980, pp. 13-23.
9. M. Shaw, "Abstraction Techniques in Modern Programming Languages," *IEEE Software*, Vol. 1, No. 4, Oct. 1984, pp. 10-26.
10. D. Gries, *The Science of Programming*, Springer-Verlag, New York, 1981, pp. 172-178.
11. R.N. Britcher and A.R. Moore, "Increased Productivity Through the Use of Software Engineering in an Industrial Environment," *Proc. Fifth Int'l Computer Software and Applications Conf. (COMP-SAC)*, CS Press, Los Alamitos, Calif., 1981, pp. 199-205.

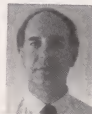
Moving?

PLEASE NOTIFY
US 4 WEEKS
IN ADVANCE

IEEE Service Center
445 Hoes Lane
Piscataway, NJ 08854

ATTACH
LABEL
HERE

- This notice of address change will apply to all IEEE publications to which you subscribe.
- List new address below.
- If you have a question about your subscription, place label here and clip this form to your letter.



Robert N. Britcher has worked for IBM's Systems Integration Division for 19 years, primarily on the automation of air traffic control systems. He has written about several aspects of software, including design, standards, metrics, maintenance, and software systems engineering. Britcher received the BA degree in chemistry from Gettysburg College. He is a member of the IEEE Computer Society.

Readers may write to the author at IBM, Systems Integration Division, 9231 Corporate Blvd., Rockville, MD 20850.

Name (Please Print) _____

New Address _____

City _____

State/Country _____

Zip _____



GREAT IDEAS HAVE SIMPLE BEGINNINGS.

Our relentless desire to communicate with one another began with the placement of primitive etchings on a wall.

Since those humble beginnings, methods of communicating have undergone countless, dramatic evolutionary changes. What has remained constant, certainly with people at Motorola Cellular, is the desire to pursue new 21st Century technology.

Our growth since the first cellular prototype in the early '70s has been nothing short of spectacular. And, as our business grows, so does our need for talented professionals who can help transform 21st century research and development technologies into high quality customer products. Ours is an unusual environment in which participation in the management of work is provided at every level, enabling each individual to closely identify with the success of the product.

Our current openings require individuals with appropriate background in one of the following disciplines.

SOFTWARE ENGINEERING

Projects involve design and development of data communication protocols in an integrated voice and data environment.

Requirements: 3-7 years' experience in software development using structured software analysis and design methodology; communication architectures and protocols. Real-time programming in UNIX® and "C" preferred. Ada and workstations experience a plus.

COMPUTER ENGINEERING

Positions will interest candidates through the engineering management level, and focus on the development of state-of-the-art products/systems using 68000, 68020, workstations, etc.

Requirements: Experience in any of the following: real-time software design; microprocessor hardware (680X0); C/UNIX®; microprocessor applications and interfaces; common channel signaling No. 7; X.25 protocols; CAE and PCM; digital testing; digital and analog circuit design; firmware development; digital communications and DSP. Telephony experience preferred.

We are seeking risk takers who want to shape history, rather than merely to be swept up as a part of it. If you have the technical savvy, the entrepreneurial bent, and the enthusiasm for excellence, we can provide you with the environment and the tools so that you, too, can make your mark on the future.

Our compensation package recognizes and rewards level of experience, and is supplemented by a most generous benefit program. Indicating position of interest, submit resume with salary details to: Staffing Manager, Dept. #SS8087, Motorola Inc., Cellular Group, 1501 West Shure Drive, Arlington Heights, IL 60004. An equal opportunity/affirmative action employer.

For your convenience, fax your resume over our automatic "24-HOUR FAX HOT LINE" by dialing 1-312-632-5717



MOTOROLA INC.

Cellular Group

Advanced Electronics for a More Productive World.

A Taxonomy for Computer Architectures

David B. Skillicorn
Queen's University at Kingston

Flynn's classification of architectures does not discriminate clearly between different multiprocessor architectures. Since the number of multiprocessor architectures has increased substantially, it has become important to find a useful way to describe them—a way that distinguishes those that are significantly different while revealing the underlying similarities between apparently divergent designs.

This article presents a taxonomy for computer architectures that extends Flynn's, especially in the multiprocessor category. It is a two-level hierarchy in which the upper level classifies architectures based on the numbers of processors for data and for instructions and the interconnections between them. A lower level can be used to distinguish variants even more precisely; it is based on a state machine view of processors. I suggest why taxonomies are useful in studying architecture and show how mine applies to a number of modern architectures.

There has been a rapid growth in the number of proposed and constructed architectures over the past 10 years. Many of these have been highly parallel in nature because the applications envisaged demand increased performance that uniprocessor systems cannot continue to provide. The relationships between the large number of parallel architectures are

This taxonomy extends Flynn's classification of architectures to be more discriminating. In particular, the growing variety of multiprocessors can be categorized and related.

difficult to appreciate because of the lack of a sufficiently rich taxonomy. Only one formal taxonomy, due to Flynn,¹ is in general use and it does not begin to differentiate the large number of possible multiprocessor architectures. Flynn classifies architectures by the number of instruction and data streams that they can process simultaneously. The categories are:

- single instruction, single data (SISD)

- single instruction, multiple data (SIMD)
- multiple instruction, single data (MISD)
- multiple instruction, multiple data (MIMD)

In this article, I present a classification scheme, or taxonomy, that extends Flynn's to make it more discriminating. It is based on a functional view of architecture and on information flow between units. I will show that this scheme classifies existing architectures well and also suggests new possibilities.

Taxonomies are important in classifying the world. Good taxonomies should group together those objects that are strongly related in important rather than unimportant ways. For example, biologists group whales with other mammals (warm-blooded, live-bearing) rather than with fish (swim in water) because their relationship to mammals is more important—if less obvious—than their relationship to fish.

Biologists have also shown that realistic taxonomies are hierarchically structured. My taxonomy is a two-level structure that subsumes Flynn's classification. Gross discriminations can be made at the higher level, with finer distinctions made at the second level.

Other classification schemes have been described in the literature. Feng's classifi-

cation is perhaps the best known.² It is a performance-oriented classification that describes the parallelism of the set of processors in a machine in terms of the number of bits that can be processed simultaneously. Machines are described by a pair, the first element giving the width of a word, and the second the depth—the number of words that can be operated on simultaneously. It does enable comparisons of performance between widely differing architectures, but precisely because it relates diverse architectures it does not highlight their differences.

More descriptive classifications also exist. An early classification of Reddi and Feurstel³ uses physical organization, information flow, and representation and transformation of information as the basis for classification. Information flow can be a powerful and general way of describing architectures, but their other attributes are very implementation oriented. Another popular descriptive classification is due to Händler.⁴ His classification describes architectures by giving the number of processors and how they can be pipelined together, and the width and depth of the arithmetic/logic units. While this works well for describing conventional vectorized machines, it is not clear how to generalize it to newer multiprocessor architectures.

Reasons for an architectural model.

There are three reasons for classifying architectures.

The first is to understand what has already been achieved. Until the past two decades, almost all computer systems used a von Neumann architecture. Even when the underlying hardware began to contain a limited amount of parallelism (for example, systems such as the CDC6600), it was generally concealed from the user. However, since that time, the growth in systems with different kinds of parallelism has been explosive, and it is not at all clear which architectures have the best prospects for the future.

The second reason for having a classification of architectures is that it reveals possible configurations that might not have otherwise occurred to a system designer. Once existing systems have been classified, the gaps in the classification can suggest other possibilities. Of course, not all such possibilities will be improvements. They may already have been thought of and discarded as worthless. However, until the search space has been clearly delimited, there can be no assurance that all the possibilities have been examined.

The third reason for a classification scheme is that it allows useful models of performance to be built and used. I have already mentioned that a drive for greater performance lies behind almost all new architectural ventures. A good classification scheme should reveal why a particular architecture is likely to provide a performance improvement. It can also serve as a model for performance analysis.

Approaching classification. If we consider the whole question of computation and machines that provide it, it seems helpful to think at several different abstraction levels. At the highest, most abstract level, we can consider the *model of computation* being employed. Most computers still employ a model of computation called the von Neumann model. In this view of computation, primitive operations consist of the usual operations of mathematics: addition, multiplication, comparison, and so on (in suitably restricted domains). Primitive data, however, are kept in named locations, and correct computational behavior is enforced by the programmer who describes the exact order in which computations are to be performed. In fact, the execution order is over-constrained in the sense that there are other orderings of all but trivial programs that compute the same result; but there is no way to express this in the computational model.

The von Neumann model of computation seems natural to those whose programming experience was developed using languages that support it. However, it simply does not contain any concept of computing more than one thing at once (parallelism), nor does it contain any way in which the possibility of ordering computations dynamically can be expressed.

It is not surprising that the first new computation models developed were extensions to the von Neumann model, to which concepts such as communication primitives were added. Later, other, more unusual, computation models—such as dataflow and graph reduction—were developed.

As we move to a more detailed level, we can consider *abstract machines* as the implementation of models of computation. The mapping from model of computation to abstract machine is not necessarily a 1-to-1 mapping; some models of computation can be implemented by more than one type of abstract machine, although there is usually a better fit for some than others. An abstract machine captures the essence of a particular archi-

ture form, without distinguishing different technologies, implementation sizes, or the like. For instance, the traditional von Neumann abstract machine can be evoked for most programmers by a summary such as program counter, addressing modes, condition codes, control unit, and arithmetic/logic unit. This abstract machine does not distinguish between the different von Neumann implementations that exist, nor does it distinguish between reduced-instruction-set computers and complex-instruction-set computers. It is nevertheless a useful level at which to consider architectures, and it is this level that forms my taxonomy's highest level.

At the next, more detailed, level we can consider *machine implementations*. This need not necessarily concern only the technology used to implement the machine. Rather, this level represents one definition of computer architecture, the picture of the machine presented to an assembly language programmer. (This definition is itself embedded in the world of von Neumann machines, since in many other architectures it is not clear what an assembly language programmer is, let alone what his or her view of the machine would be.) This level corresponds to the second level of my taxonomy. One could also envisage a level below this in which details of the physical implementation are used to distinguish between different processors.

I have not discussed the role of programming languages in this classification because, in some sense, the language used on a particular machine corresponds to the model of computation that it uses. If this is not the case, then there is a very bad "fit" between language and implementation and, since our goal is (implicitly at least) performance, this sort of situation can be ignored. Of course, when the von Neumann machine was all that there was, many different languages were used on the same machine. However, there is a growing trend towards building the architecture and the language together, and there are good reasons for that trend. Even for the von Neumann architecture, it is clear that Algol-like languages are a better "fit" than, say, Lisp.

Functions of a computer architecture. In my classification of computer architectures, I use four types of functional units from which any abstract machine can be constructed. These are

- An instruction processor, a functional unit that acts as an interpreter

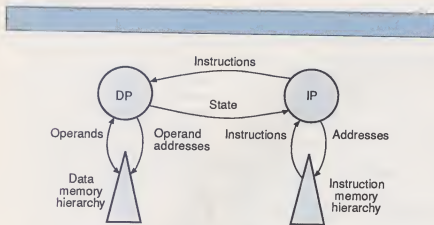


Figure 1. Arrangement of the von Neumann abstract machine functional units.

for instructions, when such things explicitly exist in the model of computation.

- A data processor, a functional unit that acts as a transformer of data, usually in ways that correspond to basic arithmetic operations.
- A memory hierarchy, an intelligent storage device that passes data to and from the processors.
- A switch, an abstract device that provides connectivity between other functional units.

In the next section, I illustrate how these functional units capture the behavior of an abstract machine, using the von Neumann architecture as an example.

The von Neumann abstract machine

The von Neumann abstract machine consists of a single instruction processor (IP), a single data processor (DP), and two memory hierarchies. The switch functional unit plays no role in describing the von Neumann abstract machine.

These functional units are arranged as shown in Figure 1. The instruction processor connects to one of the memory hierarchies from which it can fetch instructions. To do so, it must provide the memory hierarchy with a label (address) that identifies the instruction to be fetched. It also connects to the data processor, to which it sends information about operations to be performed and the labels of the values on which they are to be performed, and from

which it receives state information (condition codes).

The functions of the instruction processor are to

- (1) Determine the label of the next instruction to be executed, using information from its own internal storage and the state information conveyed to it by the data processor.
- (2) Instruct the memory hierarchy to provide the instruction with that label.
- (3) Receive the instruction and decode it. This involves determining which operation, if any, the data processor will be required to do to "execute" the instruction.
- (4) Inform the data processor of the operation required.
- (5) Determine the labels of the operands and pass them to the data processor.
- (6) Receive the state information from the data processor (after completion of the operation).

Any architecture with an instruction processor will require it to carry out steps such as these. There is no presumption, in the description, that these steps need be carried out in the order given or that certain steps may not be skipped. At the first level of the taxonomy, only the existence of certain functional units, connected in a particular way, is specified.

For the von Neumann machine, the order of operations is usually that given above. It is called the instruction execution sequence. If we wish to indicate the precise

order of the steps, then we move to the second level of the taxonomy and represent these various operations as states in a state diagram. At this second level, the instruction processor can be represented as in Figure 2, in which each state represents an operation and the arrows represent the dependencies between states. The dotted lines represent communication between the instruction processor and the other functional units in the system. If we visualize a "token" that can be placed on a state, then this state diagram—together with the token's location—gives a complete description of what the instruction processor is doing at any given moment. The token then follows the arrows between states, circling the loop once for each executed instruction.

We can continue to differentiate machines that operate according to the state diagrams given above by providing implementation level details. These might include exact mechanisms or timings for each state, or the way in which logical entities are mapped to physical ones. One such example is the following: Most von Neumann machines implement the selection of the next instruction using a program counter updated by the length of the current instruction (ignoring branch instructions). However, in some systems, the address of the next instruction is included in the instruction itself. Neither of these strategies is more von Neumann than the other. They are implementation choices and should be indistinguishable at the abstract machine level.

The data processor carries out the following steps:

- (1) Receive an instruction type from the instruction processor.
- (2) Receive operand labels from the instruction processor.
- (3) Instruct the memory hierarchy (separate from that of the instruction processor) to provide the values of the operands.
- (4) Receive operand values from the memory hierarchy.
- (5) Carry out the required operation (the execute phase).
- (6) Return state information to the instruction processor.
- (7) Provide a result value to the memory hierarchy.

Once again, this can be represented by a state diagram consisting of a loop of states. The greatest difference between the instruction processors is that the execute phase of the data processor consists of a

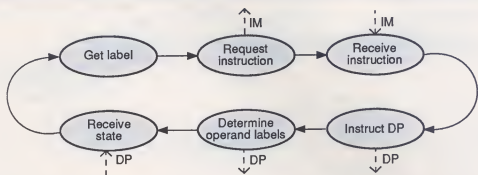


Figure 2. The internal structure of a von Neumann instruction processor.

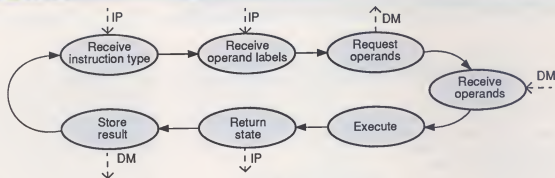


Figure 3. The internal structure of a von Neumann data processor.

potentially large number of parallel states, representing those operations that the data processor regards as primitive. The state diagram is shown in Figure 3.

Certain actions taken in both the instruction processor and data processor must be synchronized. Places where this occurs, shown using dotted lines in the state diagrams, represent communication between the processors. Their behavior follows: A token must be present on the state at the beginning and end of communication path for communication to occur. Neither token can leave that state until the communication is complete (that is, communication is synchronous).

A memory hierarchy is an intelligent storage device that attempts to provide the data requested by its attached processor as quickly as possible. I begin by justifying

my treatment of a storage as a hierarchy. Consider the ultimate (at least with the physics we now understand) storage device. At its center is the processor. The data are stored in a sphere around it and can be accessed at light speed. As we increase the amount of storage, we are forced to place some storage locations on the outer surface of the sphere at a greater radius from the processor. Hence, the time to access those locations must increase. The surface area at the larger radius is greater as well, so more data can be stored there.

Therefore, we can see that storage must inherently be organized in a hierarchy. Data stored close to the processor will have short access times but must always be of limited capacity. Adding capacity forces an increase in access time for some storage.

But storage with longer access times can be more plentiful. Thus, a storage system should be regarded as a pyramid or, more accurately, as a ziggurat, since the data stored are discrete and successive "layers" get larger in steps.

A memory hierarchy attempts to keep the next piece of data required by its attached processor at the top of the hierarchy where the access time is smallest. A perfect memory hierarchy would always achieve this goal. However, there is no optimal way to determine which piece of data will be required next. Hence, all memory hierarchies must approximate this goal using heuristics.

Clearly, the average access time is minimized when the data objects referenced most often are kept at the top of the hierarchy and those referenced least often

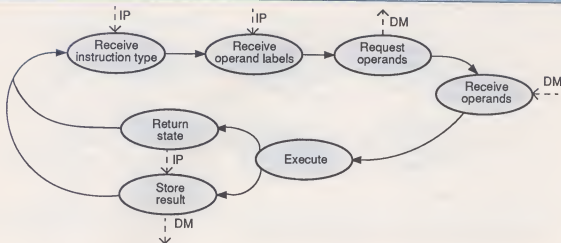


Figure 4. A faster von Neumann data processor.

are kept at the bottom. When the data accesses have some locality, either temporal or spatial, average access performance can be improved by keeping data that have been accessed recently or that are close to accessed data at the top of the hierarchy. Fortunately, almost all computation models do appear to exhibit substantial locality.

We can now see why a von Neumann abstract machine contains two memory hierarchies. Data in the instruction memory hierarchy flows towards the processor; instructions are absorbed by the instruction processor and do not themselves get modified. On the other hand, the data memory hierarchy has to manage data movement in both directions. Hence, the implementation of the two memory hierarchies may have substantially different properties. In principle, a von Neumann architecture does not distinguish between data and instructions; but, almost all real machines make it difficult for a programmer to use this equivalence, and programming environments enforce a separation. It seems useful to make the distinction in the abstract machine. Memory hierarchies are usually implemented using a single hierarchy of storage except at the top level—the cache—in which instruction and data storage are often differentiated. The cache is at the top of the memory hierarchy, above the main memory, which in turn is above disk storage (such as swap

space) and longer term storage such as tapes. I/O takes place at the bottom of the memory hierarchy. In a sense, I/O is an access to a very large storage mechanism, the outside world.

Ways to increase performance

Most of the other computational models are motivated by a desire for increased performance, so it is instructive at this point to consider how the von Neumann abstract machine could achieve better performance. There are three major ways: rearranging the state diagrams so that it takes less time for a token to circulate; allowing more than one token and, hence, more than one active step at a time; and, replicating functional units to permit concurrent activity. All three ways do increase performance, and the third is especially fruitful, leading to a large and diverse class of architectures. Notice how explicit these possible mechanisms are made by the abstract machine representation.

The first method of increasing speed—rearranging the state diagram—does not really generate a new architecture class, although my classification scheme does allow differences to be distinguished in the descriptions of the processors' state diagrams. As an example, in the data proces-

sor, the operations of informing the instruction processor of the state information derived from the result and of storing the result in the memory hierarchy are independent. They can, therefore, occur simultaneously. This leads to the revised state diagram shown in Figure 4, which shows that, regardless of the time each step takes, the new arrangement will be faster.

Pipelining. The second possible speed-up, permitting more than one token in the state diagram, is called pipelining. Suppose that the number of stages in a state diagram is n . If each stage takes the same amount of time, say t , then the time to execute one instruction is $n t$. If we allow n tokens to be present in the state diagram simultaneously, we represent n different instructions in various stages of execution. The time it takes to execute each instruction is still $n t$. However, the average rate at which instructions are completed is $1/t$ instructions per unit of time. Thus, we have achieved an n -fold speedup in our machine, as long as there are always n tokens in the state diagram. Of course, there are practical difficulties in making this work—instructions such as branches and jumps, external interrupts, and simultaneous needs to access the same data cause real pipeline performance to be much worse than this simple analysis would suggest. However, it does serve to show why pipelines were interesting to sys-

tem designers trying to improve the performance of von Neumann machines.

Pipelines have another interesting property. If we subdivide each of the states in our state diagram into two substates, each of which takes time $t/2$, and allow $2n$ tokens in the state diagram, then an instruction completes every $t/2$ time units thus increasing the completion rate of instructions to $2/t$ instructions per unit of time. Thus, we have a total speedup of $2n$. Making each stage smaller allows us to increase the completion rate, but at the expense of complicating the handling of unforeseen interactions.

Within our abstract machine model, pipelined behavior can be described without adding any new functional units. Instead, we label each functional unit as one of two types: simple or pipelined.

Array processors. Replicating functional units is the third way to improve system performance. There are many different possibilities for replication, and we begin with the simplest—array processors. A typical array processor consists of an instruction unit that broadcasts instructions to a set of slave processors. Each of the slave processors executes the broadcast instruction, interpreting the operand addresses as lying in its own memory. Instructions are usually provided to allow processors to exchange data, directly or indirectly. It is also common for each processor to have access to its own identity, allowing for different relative locations to be accessed in different processors. An array processor is straightforwardly classified as an abstract machine with a single instruction processor, a single instruction memory, multiple data processors, and multiple data memories.

Because there are multiple functional units, we must describe the ways in which they can be connected. Connections between functional units are made using abstract switches that can be implemented in different ways: by buses, dynamic switches, or static interconnection networks. Four different forms of abstract switch connect functional units together:

- **1-to-1**—A single functional unit of one type connects to a single functional unit of another. Of course, there may be more than one physical connection and information may flow in both directions, as in the von Neumann abstract machine described earlier. This is not really a switch, but I include it for completeness.
- **n -to- n** —In this configuration, the i th unit of one set of functional units connects

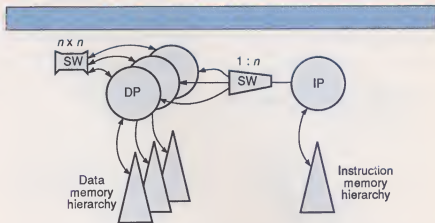


Figure 5. A Type 1 array processor.

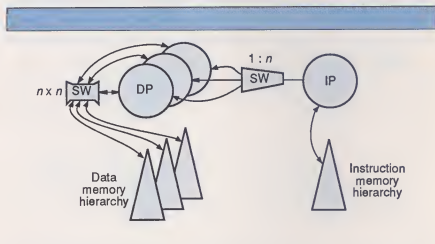


Figure 6. A Type 2 array processor.

to the i th unit of another. This type of switch is a 1-to-1 connection replicated n times.

- **1-to- n** —In this configuration, one functional unit connects to all n devices of another set of functional units.

- **n -by- n** —In this configuration, each device of one set of functional units can communicate with any device of a second set and vice versa.

All array processors have a 1-to- n switch connecting the single instruction processor to the data processors. Two different sub-families can be distinguished, based on the types of switch used to connect the data processors and data memory hierarchies.

The first kind is shown in Figure 5. Here, the data processor-data memory connection is n -to- n and the data processor-data processor connection is n -by- n . This connection scheme is used in machines such as the Connection Machine.⁵

The second type is shown in Figure 6. In this case, the data processor-data memory connection is n -by- n and there is no connection between the data processors. This form of switch is usually called an alignment network. One example of such an architecture is the Burroughs Scientific Processor.⁶

We can now introduce our taxonomy informally. We classify architectures by the number of instruction processors and

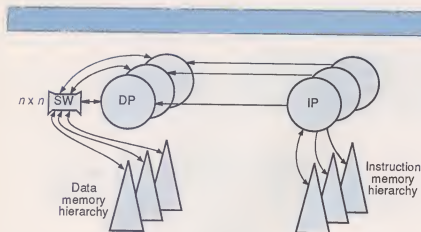


Figure 7. The abstract machine of a typical tightly coupled multiprocessor.

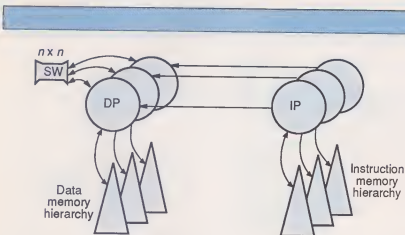


Figure 8. The abstract machine of a typical loosely coupled multiprocessor.

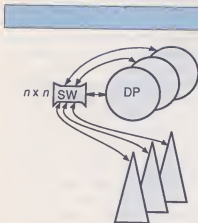


Figure 9. The abstract machine for graph reduction.

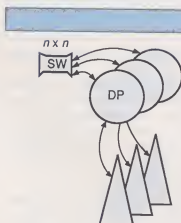


Figure 10. The abstract machine for dataflow (Dennis type).

data processors they have, by the number of memory hierarchies they have, and by the way in which these functional units are interconnected by abstract switches. Thus, a von Neumann uniprocessor can be classified as

- number of instruction processors is 1
- number of instruction memory hierarchies is 1
- switch between instruction processor and instruction memory is 1-to-1
- number of data processors is 1
- number of data memories is 1
- switch between data processor and data memory is 1-to-1
- switch between instruction processor and data processor is 1-to-1

A Type 1 array processor can be classified as

- number of instruction processors is 1
- number of instruction memory hierarchies is 1
- switch between instruction processor and instruction memory is 1-to-1
- number of data processors is n
- number of data memories is n
- switch between data processors and data memories is n -to- n
- switch between instruction processor and data processor is 1-to- n
- switch between data processors is n -by- n

We shall see more complex descriptions in the next section. As the number of functional units increases, so do the possibilities for interconnecting them.

Parallel von Neumann machines

Array processors rely on the existence of many pieces of data that can be manipulated in the same way at the same time. Many problems do not fit this paradigm well. Therefore, it is natural to consider replicating the instruction processor as well as the data processor and creating multiple control "threads." This is the approach taken in parallel von Neumann machines. Two very different architecture types result from this approach: tightly coupled systems and loosely coupled systems.

Tightly coupled systems consist of a set of processors connected to a set of memories through a dynamic switch. Any processor can access any location in the memories with about the same latency. Communication and synchronization between processes is achieved by the use of

shared variables. Examples of such machines include the BBN Butterfly, the IBM 3081 and 3090, and C.mmp. Loosely coupled systems consist of a set of processors, each with its own local memory. Communication takes place by explicit request from one processor to another over an interconnection network, by message passing, or by remote procedure calls. Examples of such machines include the Intel and NCube hypercube machines, Transputer-based systems such as Supernode (Esprit project 1085) and the Meiko MK40, as well as older systems such as CM*.

In tightly coupled systems, both data and instruction processors are replicated, but the data processors share a common data memory. The corresponding abstract machine is shown in Figure 7. I show it with multiple data memories and an n -by- n switch between data processors and data memories because it is slightly more suggestive of the usual implementation. Functionally, it is indistinguishable from a common shared memory.

Loosely coupled systems also have replicated data and instruction processors, but the switches in the data subsystem differ. A typical loosely coupled system is shown in Figure 8. The connection between data processors and data memories is n -to- n , and there is an n -by- n connection between the data processors.

Machines using other models of computation

Von Neumann machines are all based on a model of computation with threads of instructions executed sequentially, except where the order is explicitly altered. I have already commented that such ordering is over-constrained. This difficulty gets worse when computation models with multiple threads of control are used, since programmers must consider not only the ordering of instructions in a particular thread but the possible orderings of instructions in different interacting threads. It is not surprising that those involved in designing highly parallel machines have examined other models of computation that do not have this awkward property. These new models of computation are characterized by a complete absence of programmer description of the order of evaluation, other than an order implied by data dependencies. This allows any possible evaluation order to be consid-

ered for execution, and the one with the greatest performance can be selected by the compiler or the runtime environment. Models of computation with this property are usually expressed in functional languages.

Graph reduction machines. In graph reduction, a computation is encoded as a graph of functions and arguments that (recursively) have the property that the root node is an Apply, the left subtree is the description of a function, and the right subtree is the description of an argument.

The description of a function or an argument can be either a value or a description of how to compute the value. If both the function and the argument have already been evaluated, then the subtree—called a redex—is available for execution. When it has been executed, its value overwrites the subtree. If either has not yet been evaluated, then some subtree must be a redex or the computation cannot proceed. Thus, in general, redexes available for execution are found at the leaves of the tree. Typically, multiple redexes are available for evaluation at any moment, so multiple processors can be involved in evaluation simultaneously.

The abstract machine for graph reduction is shown in Figure 9. It differs from the abstract machines of the von Neumann type primarily in the absence of an instruction unit and instruction memory. The graph being reduced plays the role of both instructions (in its structure) and data (in its content). The data processors and their associated data memories are like those of a tightly coupled multiprocessor, which is not surprising since the graph is a large shared-data structure.

Since there is no instruction processor to provide data labels to the data processor, it must generate them itself. There are two equivalent ways to model this. The first is to provide the data processor with a selector that selects any available redex from the data memory. The other is to regard the data memory as actively pushing the available redexes towards the data processor.

Work on graph reduction has mainly taken place in Europe and the United Kingdom and has been very successful. Several machines have been designed or built. Some examples are Alice⁷ and Flagship.⁸

Dataflow machines. Dataflow machines are another class of machine that does not have an instruction sequencing mecha-

nism, other than that implied by data dependencies. Its model of computation is a graph with directed edges, along which data flows, and vertices representing operations that transform the data. Certain edges have a vertex at only one end; these represent the inputs or outputs of the computation. A vertex, or operator, fires according to some firing rule specified by the particular form of dataflow. In general, a firing absorbs a datum from each input arc and produces a datum that then departs along the outgoing arc(s).

Most proposed dataflow machines are based on a ring consisting of a matching store (where data values wait until a complete set of operands is present), a memory containing the operators, and a set of processing elements that execute the operators, producing result values that flow around the ring to the matching store.

The abstract machine for a dataflow processor can take two forms, corresponding to the two possible forms of data processor arrangement in an array processor. In the first form, all of the data memory is equally visible to all processors. This approach has been taken in the Manchester Dataflow Machine⁹ and in Arvind's machine.¹⁰ This form of processor is functionally identical to parallel graph-reduction machines and, hence, it has the same abstract machine diagram (Figure 9). In the second form, shown in Figure 10, each processor has its own local memory, and data values needed by other processors flow across the inter-data-processor switch. The second form is perhaps more intuitive; in the limit, each processor executes only a single operator. As before, a data processor can be modelled as having a selector or as being sent executable operators by a data memory.

The formal model

Considering the major types of processors that exist today has shown some of the properties that must be captured by a formal classification scheme. My scheme consists of two levels of increasing detail and discrimination. The first level describes an architecture by specifying

- the number of instruction processors,
- the number of instruction memories,
- the type of switch connecting IPs to instruction memories,
- the number of data processors,
- the number of data memories,
- the type of switch connecting DPs

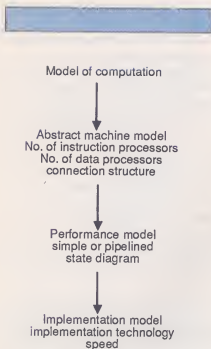


Figure 11. The classification method.

- and data memories,
- the type of switch connecting IPs and DPs, and
- the type of switch connecting DPs to DPs.

The first level refines Flynn's classification by expanding the classes of SIMD and MIMD into a set of subclasses that capture important existing architectures.

The second level allows further refinement by describing whether or not the processors can be pipelined and to what degree, and by giving the state diagram behavior of the processors. A third level describing implementation details is also possible. The approach is illustrated in Figure 11.

As we have seen, this classification scheme captures the differences between architectures that are significantly different, while ignoring differences that are primarily a matter of implementation. Table 1 illustrates the set of simple architectures possible under the assumptions that the number of memories matches the number of processors for each

subsystem. Although there may be interesting architectures for which these assumptions do not hold, we will ignore them in the interest of a clear exposition.

Classes 1 through 5 are the dataflow/reduction machines that do not have instructions in the usual sense. Class 6 is the von Neumann uniprocessor. Classes 7 through 10 are the array processors. Note that classes 5 and 10 represent extensions to the common architectures of their respective types in that they have a double connectivity for their data processors.

Classes 11 and 12 are the MISD architectures. Although these classes have been treated as an aberration, there are languages for which this type of execution is appropriate. For example, in NIAL (Nested Interactive Array Language),¹¹ it is possible to write

$[f \ g \ h] \ x$

that is the parallel application of functions f , g , and h to x . I am not aware of any existing architecture of this type, but the con-

Table 1. Possible architectures.

Class	IPs	DPs	IP-DP	IP-IM	DP-DM	DP-DP	Name
1	0	1	none	none	1-1	none	reduct/dataflow uniprocessor
2	0	n	none	none	n-n	none	separate machines
3	0	n	none	none	n-n	n × n	loosely coupled reduct/dataflow
4	0	n	none	none	n × n	none	tightly coupled reduct/dataflow
5	0	n	none	none	n × n	n × n	
6	1	1	1-1	1-1	1-1	none	von Neumann uniprocessor
7	1	n	1-n	1-1	n-n	none	
8	1	n	1-n	1-1	n-n	n × n	Type 1 array processor
9	1	n	1-n	1-1	n × n	none	Type 2 array processor
10	1	n	1-n	1-1	n × n	n × n	
11	n	1	1-n	n-n	1-1	none	
12	n	1	1-n	n × n	1-1	none	
13	n	n	n-n	n-n	n-n	none	separate von Neumann uniprocessors
14	n	n	n-n	n-n	n-n	n × n	loosely coupled von Neumann
15	n	n	n-n	n-n	n × n	none	tightly coupled von Neumann
16	n	n	n-n	n-n	n × n	n × n	
17	n	n	n-n	n × n	n-n	none	
18	n	n	n-n	n × n	n-n	n × n	
19	n	n	n-n	n × n	n × n	none	Denelcor Heterogeneous Element Processor
20	n	n	n-n	n × n	n × n	n × n	
21	n	n	n × n	n-n	n-n	none	
22	n	n	n × n	n-n	n-n	n × n	
23	n	n	n × n	n-n	n × n	none	
24	n	n	n × n	n-n	n × n	n × n	
25	n	n	n × n	n × n	n-n	none	
26	n	n	n × n	n × n	n-n	n × n	
27	n	n	n × n	n × n	n × n	none	
28	n	n	n × n	n × n	n × n	n × n	

cept is not completely ridiculous.

Classes 13 through 28 are the multiprocessors of various kinds. Classes 13 through 20 are more or less conventional multiprocessors with different forms of connection structure. Classes 21 through 28 are more exotic architectures in which the connections between instruction processors and data processors is n -by- n . These classes appear to be completely unexplored.

Architectures that have an n -by- n connection between the instruction processors and instruction memories are those in which the decision about the processor to execute a particular instruction is made late (that is, just before execution). Dataflow and graph reduction architectures often have this property, but the only von Neumann machine of which I am aware that behaves this way is the Heterogeneous Element Processor.¹²

Some existing von Neumann machines have more than a single data processor. For example, several high performance pipelined systems (Cray I, Cyber 205) have a data processor for vector instructions and another for scalar instructions. Some have a separate scalar instruction processor as well. Architectures of this type do not fit into the simple scheme outlined above, but they can be easily represented by including the vector processing units with the data processors (number of data processors is $1 + m$).

An example classification for an unusual architecture. I conclude by illustrating the complete description of an unusual processor using my classification. I have chosen the Flagship architecture, a loosely coupled graph reduction system.

The Flagship machine consists of a set of processors, each with its own local memory. The graph to be reduced is partitioned statically, and these partitions are allocated to the local memories of processors. Each processor selects possible parts of its own subgraph for reduction. If the required objects are local, the reduction proceeds and the subgraph is replaced by the reduced value. If some required part of the subgraph is not local (for instance, because it is shared among several different subgraphs), then a request for its value is sent over an interconnection network to the processor in whose local memory the subgraph resides. The value is eventually supplied by the remote processor. A processor that has completely reduced its subgraph can demand more work from other processors.

COMPUTER SECURITY SCIENTISTS

Tomorrow's Computing Technology is Today's Challenge

—at—
I D A

Some of the nation's most exciting developments in software technology, supercomputer architecture, AI, and expert systems are under scrutiny right now at the Institute for Defense Analyses. IDA is a Federally Funded Research and Development Center serving the Office of the Secretary of Defense, the Joint Chiefs of Staff, Defense Agencies, and other Federal sponsors.

IDA's Computer and Software Engineering Division (CSED) is seeking professional staff members with an in-depth theoretical and practical background in the area of Computer Security. Tasks include efforts on both the design/development of techniques to assess and assure security and providing advice to DoD decision makers on appropriate and feasible policy regarding security. Specific desired skills and interests include:

- Formal verification, with emphasis on the Ada language
- Secure kernels and reference monitors
- Security in multiprocessor systems
- Fault-tolerance in secure systems
- Operating system, data base and network security criteria
- Testing and evaluation

Specialists in other areas of Computer Science are also sought: **Software Engineers, Distributed Systems, Artificial Intelligence and Expert Systems, and Programming Language Experts.**

We offer career opportunities at many levels of experience. You may be a highly experienced individual able to lead IDA projects and programs . . . or a recent MS/PhD graduate. You can expect a competitive salary, excellent benefits, and a superior professional environment. Equally important, you can expect a role on the leading edge of the state of the art in computing. If this kind of future appeals to you, we urge you to investigate a career with IDA. Please forward your resume to:

**Mr. Thomas J. Shirhall
Manager of Professional Staffing
Institute for Defense Analyses
1801 N. Beauregard Street
Alexandria, VA 22311**

An equal opportunity employer.
U.S. Citizenship is required.



ENTER THE 1988



Gordon Bell Awards

The Gordon Bell Awards recognize achievements in large-scale scientific computing. Awards of \$1,000 each will be given in two of three categories:

- **Performance.** A winning program will run faster than comparable engineering or scientific applications.
- **Price/performance.** A winning program must show that its performance divided by system cost is less than comparable applications.
- **Compiler parallelization.** A winning compiler/application must generate the greatest speedup.

The 1987 winners achieved speedups of 400 to 600 with 1,024 processors. Entries must be received by Jan. 15, 1989. IEEE Software administers the awards. For rules, write Gordon Bell Awards, 10662 Los Vaqueros, Los Alamitos, CA 90720; (714) 821-8380.



GAO....WHERE YOU CAN MAKE A DIFFERENCE

Do you have what it takes to make technical assessments of ADP, telecommunications, or information resources projects?

We are looking for specialists who can evaluate federal automated data processing, communications, and information management systems. You should have experience managing in one of the following areas:

- ADP Systems Development
- Telecommunications
- Capacity Management
- Database Administration

The U.S. General Accounting Office (GAO) is an independent, nonpartisan organization in the legislative branch of the federal government, charged with evaluating federal programs to ensure that they are carried out efficiently, effectively, and economically. Salaries start at \$39,501 and are commensurate with your experience. To see if your background meets our future program needs, please submit a resume or SF-171 (application for federal employment) to the following address:

U.S. General Accounting Office
Office of Recruitment - IE, Room 4650
441 G Street, NW
Washington, DC 20548

Equal Opportunity Employer U.S. Citizenship Required

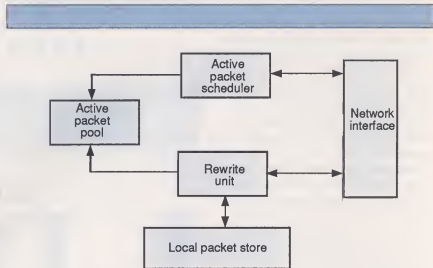


Figure 12. Conventional block diagram of the Flagship architecture.

Graphs are represented as linked structures of nodes called (somewhat inappropriately) packets. A node can be in one of three states: not ready to be reduced (because some subgraph has not yet been reduced), in the process of being reduced, or waiting for some remote object that it needs to continue reduction. This status is recorded within each packet.

The classification of this architecture at the first level is type 3 in Table 1. It corresponds to the architecture shown in Figure 10.

Now, consider the second level classification. The processor architecture in a conventional block diagram is shown in Figure 12. From this conventional description of the processor and its behavior, we can construct a state diagram for the data processor, shown in Figure 13. The way in which the Flagship memory devices form a memory hierarchy is shown in Figure 14. This more detailed description shows that the taxonomy can describe unusual architectures as well as conventional ones, and that such descriptions reveal the internal structure.

between architectures where they really significantly differ, and reveals underlying relationships as well (for example, showing the close relation between graph reduction and dataflow architectures). The taxonomy also suggests a number of unexplored possible architectures that might be fruitful places to look for new and innovative machine designs.

The taxonomy is divided into two levels. At the highest level, architectures are distinguished by the number of instruction processors, the number of data processors, the number of memory hierarchies they contain, and by the way in which these devices are connected. At this level, there are 28 simple architectures. At the lower level, further discriminations can be made by describing whether or not each of the processors is pipelined and by giving its internal functional structure by a state diagram. A further level, giving implementation details, is also possible. The emphasis throughout is on capturing the functional behavior of architectures rather than the exact way in which they carry out a set of tasks.

A taxonomy must shed light on what is already known and help in assimilating new understanding, if it is to be useful. I believe that my taxonomy is a natural extension of Flynn's work—more complex because the possibilities have grown, but still simple and straightforward enough to be used as an intellectual tool for understanding and as an engineering tool for design. □

I have presented a classification scheme for architectures that is considerably more discriminating than those presently in use. It extends Flynn's popular classification by increasing the discrimination between different kinds of parallel architectures. I have shown that my taxonomy captures the distinctions

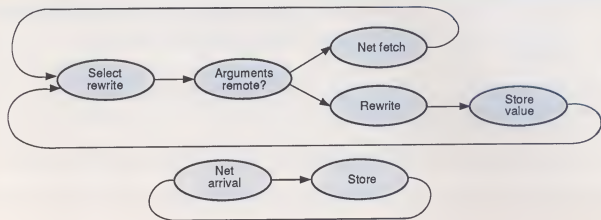


Figure 13. State diagram for the Flagship data processor.

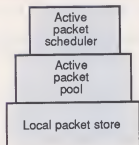


Figure 14. How Flagship memory devices form a memory hierarchy.

Acknowledgments

This work was supported by the Natural Sciences and Engineering Research Council of Canada. David Barnard and the anonymous referees made several suggestions that improved the presentation.

References

- M.J. Flynn, "Some Computer Organizations and Their Effectiveness," *IEEE Trans. Computers*, C-21, No. 9, Sept. 1972, pp. 948-960.
- T.Y. Feng, "Some Characteristics of Associative/Parallel Processing," *Proc. 1972 Sagamore Computing Conf.*, Aug. 1972, pp. 5-16.
- S.S. Reddi and E.A. Feurstel, "A Conceptual Framework for Computer Architecture," *Computing Surveys*, Vol. 8, No. 2, June 1976, pp. 277-300.
- W. Händler, "The Impact of Classification Schemes on Computer Architecture," *Proc. Int'l Conf. on Parallel Processing*, Aug. 1977, pp. 7-15.
- W.D. Hillis, *The Connection Machine*, MIT Press, Cambridge, Mass., 1985.
- D.J. Kuck and R.A. Stokes, "The Burroughs Scientific Processor," *IEEE Trans. Computers*, Vol. C-31, No. 5, May 1982, pp. 363-376.
- J. Darlington and M. Reeve, "Alice—A Multiprocessor Reduction Machine for the Parallel Evaluation of Applicative Languages," *Proc. 1981 ACM Conf. Functional Programming Languages and Computer Architecture*, 1981, pp. 65-75.
- I. Watson et al., "Flagship: A Parallel Architecture for Declarative Programming," *Proc. 15th Ann. Int'l Symp. on Computer Architecture*, May 1988, pp. 124-130.
- J. Gurd and I. Watson, "Data Driven Systems for High Speed Parallel Computing: Part 1: Structuring Software for Parallel Execution; Part 2: Hardware Design," *Computer Design*, June and July 1980, pp. 91-100, 97-106, respectively.
- A. and R.S. Nikhil, "Executing a Program on the MIT Tagged Token Dataflow Architecture," *Proc. Parallel Architectures and Languages Europe (PARLE) Conf.*, Springer-Verlag, Berlin, Lecture Notes in Computer Science, June 1987, pp. 1-29.
- M.A. Jenkins, J.I. Glasgow, and C. McCrosky, "Programming Styles in NIAL," *IEEE Software*, Vol. 3, No. 1, Jan. 1986, pp. 46-55.
- Denelcor Inc., *Heterogeneous Element Processor: Principles of Operation*, Apr. 1981.



David B. Skillicorn is an associate professor in the Department of Computing and Information Science at Queen's University at Kingston, located in Kingston, Ontario, Canada. He is currently a visiting researcher in the Programming Research Group at the University of Oxford in England. His research interests include multiprocessor architectures and functional languages and how they interact to produce high-performance parallel systems. He is a member of the IEEE Computer Society and the Association for Computing Machinery. Skillicorn received the BSc degree from the University of Sydney in 1978 and the PhD from the University of Manitoba in 1981.

Readers may write to Skillicorn at the Dept. of Computing and Information Science, Queen's University at Kingston, Kingston, Ontario, Canada K7L 3N6.



The Fifth International Conference on Data Engineering

February 6-10, 1989
Los Angeles Airport Hilton and Towers
Los Angeles, CA

General Chairperson: John Carlis, University of Minnesota

Program Chairperson: Richard L. Shuey, Rensselaer Polytechnic Institute

SCOPE

Data Engineering is concerned with the semantics and structuring of data in information systems design, development, management and use. It encompasses both traditional applications and issues, and emerging ones. The purpose of this conference is to provide a forum for the sharing of practical experiences and research advances from an engineering point of view among those interested in automated data and knowledge management. We expect that this sharing will enable future information systems to be more efficient and effective, and future research to be more relevant and timely.

INVITATION

This conference will prove well worth attending. There will be three days for the presentation of 72 papers in general sessions and five panels in three tracks. Also, keynote speakers will address attendees each day. A fourth track of vendor presentations will complement the research tracks and ten full-day and half-day tutorials will provide an opportunity for intensive exposure to specific data engineering topics. In addition, time will be available for questions and discussions. Come and join us!

The tutorials scheduled are entitled:

- Number 1: An Introduction to Object-Oriented Database Systems
Monday, February 6th—8:30am-5:00pm
- Number 2: Expert Database Systems—A Logical Approach
Monday, February 6th—8:30am-5:00pm
- Number 3: PC-Based Database Management Systems
Monday, February 6th—7:00-10:15pm
- Number 4: Integrating Heterogeneous Distributed Databases:
Requirements, Concepts and Solutions
Monday, February 6th—7:00-10:15pm
- Number 5: Artificial Neural Networks and Applications
Monday, February 6th—7:00-10:15pm
- Number 6: Extracting Knowledge from Data and Implications for
the Architecture of Future Knowledge Base Management Systems
Tuesday, February 7th—7:00-10:15pm
- Number 7: Recursion in Database Systems
Wednesday, February 8th—7:00-10:15pm
- Number 8: Communications Technology for Data Engineers
Friday, February 10th—8:30am-12:00noon
- Number 9: Extensible Database Systems
Friday, February 10th—8:30am-12:00noon
- Number 10: Engineering Data Management
Friday, February 10th—8:30am-12:00noon

Conference general sessions and panels will be held Tuesday, February 7th thru Thursday, February 9th. Papers related to the following topics will be presented:

- Object-Oriented Database Issues
- Federated Database Systems
- Query Processing Techniques
- Non-Traditional Database Systems
- Database Machines
- Panel: Information System Disasters and Their Prevention
- Applications
- Communications Issues
- Data Modeling
- Panel: Does Loose Coupling of AI-DBMS Stand a Chance?
- Distributed Database Issues
- Data Structures and File Organizations
- Data Security
- Recursive Queries
- Applications of AI in Database
- Panel: Object-Oriented Databases—Evolution or Revolution?
- Complex Objects
- DBMS Performance and Implementation Issues
- Panel: Parallel Data Architectures: Electronics, Optics, Neural, Objects?
- Performance Issues for Parallel and Distributed Systems
- Handling Partitioning in Distributed Database Systems
- Query Processing Techniques
- Spatial and Geographic Data
- Database Design
- Deductive Databases
- Panel: Issues of Data Engineering

The conference will be held at the **Los Angeles Airport Hilton and Towers**, 5711 W. Century Blvd., Los Angeles, CA 90045-5631, 213-410-4000. For reservations call 1-800-HILTONS. Please identify yourself as a Data Engineering conference attendee to receive the special rate of \$78.00, single or double.

For a full Data Engineering advance program, please contact the IEEE Computer Society, 1730 Massachusetts Avenue, NW, Washington, DC 20036, (202) 371-1013.



IEEE COMPUTER SOCIETY



THE INSTITUTE OF ELECTRICAL
AND ELECTRONICS ENGINEERS, INC.

ADVANCE REGISTRATION FORM

Send form or facsimile to:

Glenda McBride
Data Engineering Conference
IEEE Computer Society
1730 Massachusetts Avenue, NW
Washington, DC 20036-1903

Please Type or Print

Name _____
Affiliation _____
Mailing Address _____
City _____ State _____ Zip _____
Country _____
Work Phone _____ Telex/fax number _____ IEEE/CS Membership Number _____

Conference Registration Fees

Advance Registration—Prior to January 9, 1989

	Member	Non-Member	Student
Conference Only	\$210.00	\$265.00	\$40.00
Half-day Tutorial	\$100.00	\$125.00	\$100.00
Full-day Tutorial	\$160.00	\$200.00	\$160.00

Late/On-Site Registration—After January 9, 1989

	Member	Non-Member	Student
Conference Only	\$255.00	\$320.00	\$40.00
Half-day Tutorial	\$120.00	\$150.00	\$120.00
Full-day Tutorial	\$200.00	\$250.00	\$200.00

Method of Payment

☐ Personal Check ☐ Company Check
☐ Visa ☐ MasterCard ☐ American Express

Card Number _____ Exp. Date _____
Signature _____

Conference Amount \$ _____
Half-day tutorial fee(s) \$ _____
Full-day tutorial fee(s) \$ _____
TOTAL ENCLOSED \$ _____

Please check the tutorial(s) you wish to attend:

- ☐ Tutorial 1—Intro to Object-Oriented Database Systems
☐ Tutorial 2—Expert Database Systems
☐ Tutorial 3—PC-Based Database
☐ Tutorial 4—Integrating Heterogeneous Distributed Databases
☐ Tutorial 5—Artificial Neural Networks
☐ Tutorial 6—Architecture of Future Knowledge Base Mgmt Systems
☐ Tutorial 7—Recursion in Database Systems
☐ Tutorial 8—Communications Tech for Data Engineers
☐ Tutorial 9—Extensible Database Systems
☐ Tutorial 10—Engineering Data Management

Note: Conference registration fee includes admission to the technical sessions, one copy of the conference proceedings (students do not receive proceedings), break refreshments, and receptions. Refunds, less a \$15.00 handling fee, will be made if requested in writing no later than January 19, 1989. Tutorials registration includes one copy of the tutorial notes.

FOR OFFICE USE ONLY

Date received _____ Check No. _____ Amount _____
PC _____ CC _____ MO _____
Remarks _____

STANDARDS

Editor: Helen M. Wood, National Oceanic and Atmospheric Administration, Rm. 1069, Federal Bldg. No. 4, Washington, DC 20233, phone (301) 763-1564.

European conformance testing program progresses, unearths problem areas

Kenneth Thompson, Commission of the European Communities

Although only about a decade old and still not without flaws, the European information technology standards conformance testing program is making genuine progress.

The European Community, as it is known today, came into existence in 1957 with the signing of the Treaty of Rome. Although the community now has a common flag—a true identity symbol—it does not yet possess a single marketplace. The EC is aiming for such harmony, but has not had as much time as the US to achieve it.

The parties to the treaty foresaw the free movement of people and the removal of barriers to trade within the European Community. Our specific interest in information technology (IT) conformance testing services (CTS) stems from the need to ensure the capability to move information throughout the European Community using modern complex information technology. The European situation is leading to some solutions different from those the US found previously.

To appreciate the activities of the EC in relationship to IT conformance testing service, it is appropriate to relate it to both the rate of progress of the underlying standardization scene and the corresponding EC legislative actions.

Only 20 years ago, IT standards were

little more than a common publicly available and useful register produced by—and for the use of—manufacturers and academic technical experts. They registered the technical aspects of common practice in the rapidly advancing IT jungle.

More recently, purchasers of IT equipment have progressively, though insufficiently, contributed to the standardization work. They have found it useful to progressively refer to the standards as a legal basis for their contracts of sale and purchase. Unfortunately, the early standards were not written for such use. In addition, the standards writing committees needed to be more specific when writing technical standards.

It became increasingly important for standards to have a clear conformance statement of the requirements to claim conformance to each of the standards, and to have an independent testing service to verify the various declarations of conformity.

The EC only began to formally look at IT in 1974 when a European council agreed on a resolution that led to launching an official program through a 1979 council decision. Since then, progress has accelerated somewhat, with issuance of a council directive in 1983 covering exchange of official standardization programs, another in 1986 on mutual recognition of telematic terminals, and a council decision in 1987 on the requirement to refer to official standards in public procurement.

The specific IT domain has received an added impetus recently as a direct result of the EC setting a global target date of 1992 for creation of its single market.

EC contribution to IT standards work. The situation in Europe is even more complicated than in the US; not only do all the interested parties wish to participate, but the European Community has to make allowances for its long internal history of cultural differences. Contributors to IT work may be classified under several broad categories,

namely governmental sectors, standards organizations, telecommunications services, manufacturers, and customers.

The parties interested in IT standardization for each category should be expected to fulfill a certain role and take a corresponding responsibility. In general terms, the governmental sectors are responsible for intercountry agreements and, in particular, the official recognition of certification schemes. Standards organizations can coordinate independent test centers into a coherent, single third-party conformance test service. Manufacturers can give their own first-party declarations of conformity and contribute enormously to guaranteed practical interoperability. Customers, both public and private, can contribute to creating a larger harmonized market for the manufacturers and for their own benefit by getting together and publishing the common and stable part of their requirements in a vehicle such as the Government Open Systems Interconnection Profiles (GOSIP).

The EC is encouraging the member state governments to collaborate with the objective of using electronic mail for an increasing amount of EC administrative business. This requires a more harmonized use of IT standards in procurement. In this context, the EC welcomes such dynamic initiatives that have led to such documentation published under the GOSIP heading—whether it be US-GOSIP or UK-GOSIP.

Contribution of EC standardization organizations. National standards organizations have played a major coordinating role via their common European associations, the European Committee for Standardization, known as CEN, and the European Committee for Electrotechnical Standardization, known as CENELEC (see Table 1). They have also had the cooperation of the European group of postal, telegraph, and telephone authorities in the European Conference of Postal and Telecommunications Administrations

Note: This report is based on a talk Kenneth Thompson presented at the Symposium on Testing for Conformance to Information Technology Standards, sponsored by the National Bureau of Standards in Gaithersburg, Maryland. Thompson, principal administrative officer of the Commission of the European Communities in Brussels, was among speakers from Europe, Japan, and the US who described current and planned initiatives in conformance testing.

Additional information on CEC activities may be obtained by contacting the Directorate—General XIII, Telecommunications, Information Industries, and Innovation, Commission of the European Communities, 200 Rue de la Loi, B-1049 Brussels, Belgium, phone 32 (2) 235-12-70 or 38-67, fax 32 (2) 235-93-79.

(CEPT), and have collectively supported standardization work on the harmonized use of international standards for Open Systems Interconnection.

CEN, CENELEC, and CEPT have established some common methods of working for the IT field, embodied in some agreed-upon statements called memoranda. The third memorandum related specifically to the coordination and mutual recognition of independent conformance testing service and test centers within the EC. This has recently led to the creation of a formal mutual recognition and conformance coordination activity called the European Committee for Information Technology Certification.

The mechanism being established is not a single European institution solution, for it only establishes the formal mutual recognition between national conformance testing activities. On the other hand, it is important to understand that this mechanism also encourages some specialization by not requiring a conformance testing center in each national member state.

Manufacturers' contributions in the EC. The manufacturers have played a major role in various alliances, from the European Computer Manufacturers Association to the Standards Promotion and Application Group. The latter, formed by 12 major Europe-based manufacturers and telecommunication operators, has contributed to interoperability, both in supporting the functional standards work of CEN, CENELEC, and CEPT and in offering some actual interoperability testing services.

Individually, manufacturers will no doubt continue to make their own declarations of conformity to the IT standards relevant to their own equipment. However, many of the EC governmental procurers believe such declarations must be backed by independent conformance testing services to raise the degree of consumer confidence to an acceptable level.

Goals, conflicts, and achievements of conformance testing services. The European Commission started its active support of CTS-type activities in 1978 on the topic of Cobol compilers. The US initiative on Cobol conformance testing was recognized as important, and the commission intervened to assist in the migration of the US Cobol compiler testing suite across the Atlantic into France, Germany, and England. Another EC initiative was started in support of the Graphics Kernel System, a standard for computer graphics, while

Pascal testing seems to have grown successfully with negligible EC support.

During the early phase, we in the commission learned as much about what to avoid as what to act on. Specifically, the EC knew

(1) Harmony was desired across the entire EC, but we faced many differences between the member states.

(2) The policy was to eliminate technical barriers to IT information exchange and interworking between EC member states, but this raised the problem of maintaining alignment with the rest of the world.

(3) The goal of building the IT testing services relatively quickly required building on existing centers of expertise, and this raised the possibility of creating monopoly or dominant-position test centers or the problem of the equivalence of different test tools.

(4) The acceleration of the work needed extra funding, and it was necessary to establish the level of financial support that should be made available.

In addition, we lacked a single European Community certification scheme and authority applicable to the IT sector. There was no single registration scheme for locating and verifying conformity of IT products.

The global problem of standardization, conformance, and certification is being addressed in parallel with that for the IT sector. The commission has been working since 1986 with member states on a global scheme of certification recognition. In that context, a drafted discussion paper (Certif. 87/19) outlined work through a three-tier structure consisting of

Tier 1: Political oversight, supervision, and guidance

Table 1. Equivalent roles of the contributor organizations.

Type of organization	Europe	United States	World
Government	Commission of the European Communities	National Institute of Standards and Technology (formerly National Bureau of Standards)	
Standards	European Committee for Standardization	American National Standards Institute	International Organization for Standardization
	European Committee for Electrotechnical Standardization		International Electrotechnical Committee
Telecommunications	European Conference of Postal and Telecommunications Administrations		International Telegraph and Telephone Consultative Committee
	European Telecommunications Standards Institute		
Manufacturers	Standards Promotion and Application Group	Manufacturing Automation Protocol	
Customers	European MAP Users Group	Technical Office Protocol	

US exporters urged to closely follow European standards developments

The Commission of the European Communities is acting swiftly to turn the 12-member organization into a unified integrated market of 320 million people by the end of 1992. EC legislation dealing with standardization is likely to have a profound effect on US exports, predicts a report released by the US Commerce Department's National Institute of Standards and Technology, formerly the National Bureau of Standards.

"I encourage the American business community to follow EC standards developments closely and exert its influence wherever possible to assure fair treatment of US products in Europe," said NIST Director Ernest Ambley.

The new NIST summary report, entitled *A Summary of the New European Community Approach to Standards Development*, recommends US business interests establish communications with their European subsidiaries, distributors, or American industry associations to obtain up-to-date information regarding the development of European directives and harmonized standards. US companies are also urged to seek and take opportunities to comment on and attempt to influence proposed European directives and standards.

The NIST report cites two significant recent changes by the EC regarding standardization activities:

- The restructuring of existing European national standardization programs into a coherent group.
- An effort to quickly develop a body of technical European standards for high-technology products and equipment such as advanced composite materials and telecommunications equipment to be sold within the member states.

The European regional standards organizations—CEN, CENELEC, and CEPT—are developing standards in consultation with the existing national standardization organizations.

In the report, NIST standards analyst Patrick W. Cooke summarized the primary US standardization concerns:

- EC standards may be framed in such a way as to exclude or hinder the entry of US products into European markets.
- Imported products from the EC are likely to become less costly due to operational economies of scale and other economic benefits from standardization.
- Currently, US exporters are not

given the opportunity to review and comment on proposed EC directives and regional standards during the development phase, preventing US positions from being considered at crucial times.

Through its Office of Standards Code and Information, NIST can assist manufacturers and exporters to obtain foreign trade-related technical standards and information on regulations and certification, or refer requesters to the appropriate foreign inquiry points. This activity is a part of the NIST effort to support the Agreement on Technical Barriers to Trade (Standards Code) of the GATT (General Agreement on Tariffs and Trade). NIST also works closely with the Commerce Department's International Trade Administration and the Office of the US Trade Representative to help exporters with standards-related trade problems.

Trade-related standards information is maintained by the NIST National Center for Standards and Certification Information. The center, which serves as a central depository and inquiry point for standards and standards-related information in the US, includes a reference collection of more than 240,000 standards, specifications, test methods, codes, and recommended practices.

The NCSCI office also maintains notifications of proposed foreign government regulations issued by signatories to the Standards Code. The notifications are published by the American National Standards Institute in the *ANSI Reporter* and by the Commerce Department in the *Commerce Business Daily*. Updated weekly, a GATT hotline provides information on the latest notifications of proposed foreign regulations. The hotline number is (301) 975-4041. Announcements of draft CEN and CENELEC standards are also published in the *ANSI newsletter*.

The new NIST report contains a list of EC contact points for European standardization organizations and US government contacts for information on various aspects of EC activities related to standardization.

A copy of the report, *A Summary of the New European Community Approach to Standards Development*, (NISTIR 88-3793-1) can be obtained by sending a self-addressed mailing label to Patrick W. Cooke, Office of Standards Code and Information, A629 Administration Bldg., NIST, Gaithersburg, MD 20899. The phone number is (301) 975-4033.

Tier 2: A European council for certification and testing

Tier 3: Sectorial committees

The IT sector must work within the general schema. The commission has been both supporting and crosslinking to the work of CEN and CENELEC, which have provided an IT sectorial policy memorandum, M-IT-03, and working schema CEN/CENELEC-CCC/I-N71.

As stated above, CEN and CENELEC are implementing their voluntary certification scheme. This is directly applicable to the private sector. The status of the scheme in the public procurement sector is being reviewed for acceptance by the member states in the Senior Officials Group for IT Standardization. It is unlikely any serious differences will emerge between the public and private sectors, since the majority of the IT conformance testing centers are integral to both. There is some urgency to complete the formal European Community Mutual Recognition Scheme.

Need to accelerate work. There is a constant need for advancing standards more quickly. The need also applies to conformance testing services, normally developed after the standard is completed. It is perfectly feasible to shorten the time needed to develop both the standards and the tests by overlapping the developments. This process also improves the development of the base standards.

Unfortunately, this overlapping process invariably introduces repetition of work since the developing test suite has to follow the corresponding development of the standard. The net result of this process is an increased use of resources and a higher budget. Since the budgets of the EC and the test laboratories involved are limited, applying limited overlapping is a possibility.

Clearly, IT standardization is a partially catalytic process largely supported by experts paid directly by the industry. The questions remain: Is the European informatics (the European equivalent for what Americans call information processing technology) community satisfied with the rate of progress? Or, is it sufficiently dissatisfied to significantly increase the available budget?

The European Community needs long-term public availability for IT conformance testing services. Consequently, the commission is keen to ensure the longer term financial viability of CTS. The financial viability of the test laboratories depends on the demand for their services, but this via-

bility also depends on the ability of the test centers to actively sell their services.

It is far from evident that all the test laboratories are in the business of active promotion and selling. Similarly, the procurement agencies must also be encouraged to require conformance of products as a condition for purchase.

Many of the existing testing laboratories have not traditionally invested their own financial resources, sought venture capital, or taken bank loans to set themselves up to offer CTS. Consequently, some test laboratories are not able to participate in the partially subsidized CTS support scheme program unless they can get other finances from their national governments.

The wide use of the same testing tools is desirable for world harmony. Unfortunately, such original work, combined

with the need for commercial survival and recovery of capital investment, can lead to market dominance by the test tool owners. Clarification of the balance between world original work and world dominance is needed.

Information technology is but a single sector in the much wider global world of standardization, conformance, and certification. It is an important sector that should strongly influence the global situation, but the IT sector must ultimately become a true subset of the global world. We have not solved all these problems.

The CTS program, while not perfect, has been a well-timed initiative by the European Community. It is stimulating progress in the standards area and exposing the problems that still need resolution.

Omnibus Trade Act expands standards bureau into National Institute of Standards, Technology

With President Reagan's signing of the Omnibus Trade and Competitiveness Act, the US Commerce Department's National Bureau of Standards has become the National Institute of Standards and Technology.

The name change emphasizes potentially far-reaching augmentation of the agency's objective, according to spokesperson Michael Baum. Under one section of the new law, known as the Technology Competitiveness Act, several new assignments were added to the traditional functions of the 87-year-old NBS designed to boost US industry in the world marketplace.

The new responsibilities build on the technical expertise of the NBS, the only federal laboratory that has the support of US industry as a specific mission.

The law authorizes NIST to "assist industry in the development of technology and procedures needed to improve quality, modernize manufacturing processes, ensure product reliability, manufacturability, functionality, and cost-effectiveness, and facilitate the more rapid commercialization, especially by small- and medium-sized companies throughout the United States, of products based on new scientific discoveries."

NIST will continue to serve as the nation's central laboratory for developing and disseminating measurement standards and scientific data for science, engineering, manufacturing, commerce, industry, and education. The agency provides various services to industry, including calibration services,

standard reference data, standard reference materials used to gauge the accuracy of measuring devices and systems, and measurement quality assurance programs. These activities will form the technical core of NIST, Baum said.

There will be no interruption in the traditional measurement services the NBS has been providing, he added.

The new NIST assignments also draw on NBS' long experience in cooperative research ventures with private industry and universities. NIST is instructed to

- create a series of "regional centers for transfer of manufacturing technology" affiliated with nonprofit institutions or organizations,
- create a program to provide assistance and make federal technology available to state and local technology programs and technology extension services,
- establish an "advanced technology program" to encourage the commercialization of new high-technology products, and
- support a Commerce Department

"clearinghouse of state and local initiatives on productivity, technology, and innovation" to provide technical and analytical help to state and local officials making decisions on technology policy.

Under the new law, NIST has until December 21, 1988, to submit a new organization plan to Congress. At present, only the program for regional manufacturing technology centers has received funding.

X3 conducts second review on draft tape cassette standard

X3, the Accredited Standards Committee on Information Processing Systems, is conducting a second two-month public review and comment period on draft proposed American National Standard X3.164-198x, Unrecorded Magnetic Tape Cassette for Information Interchange. The review period extends through December 6.

This standard presents the minimum requirements for the physical and magnetic interchangeability of a 3.81 mm (0.150 inch)-wide magnetic tape cassette recorded at 394 ft/min (10,000 fpm) between information processing systems, communications systems, and associated equipment utilizing a standard code for information interchange.

The standard refers solely to a magnetic tape cassette for digital recording and supports ANSI Serial Recorded Magnetic Tape Cassette 3.81 mm (0.150 inch)-wide Tape, Four Track, 200 b/min (5,120 bpi) GCR, Streaming Mode, X3.158-198x.

Copies of the draft standard can be obtained from Global Engineering Documents by dialing (800) 854-7179 (in the US) or (714) 261-1455 (outside the US). The US price is \$25, and the international price is \$32.50.

Remote database access task group approved

Accredited Standards Committee X3 on Information Processing Systems has approved a new task group, X3H2.1, for remote database access. In 1987, X3 approved a project to develop an American National Standard for Remote Database Access—Service and Protocol.

X3H2 experience indicated a substantial corporate interest represented by X3H2 members, but, in many cases, different individual experts were addressing SQL and RDA issues for a single organization.

Those with a strong RDA focus concluded that a task group would provide for separate RDA participation while preserving the necessary link with the database languages expertise within the current X3H2 membership.

The first meeting of the new task group was held in September.

To join the new task group, contact Richard Gerhardt, General Motors, Mail Stop A/MD39, 30300 Mound Road, Warren, MI 48090-904.

NEWS FROM THE COMMITTEE ON PUBLIC POLICY**IEEE/US Manpower Committee seeks help on immigration reform bill***Paul J. Kostek, COPP member*

The issue of US immigration where it regards alien students and engineers has been a concern of the IEEE/US Manpower and Pensions Committee for some time. Within the IEEE Computer Society, the subject falls under the auspices of the Committee on Public Policy subcommittee on Human Resources and Immigration.

Computer Society members are asked to participate in formulation of an IEEE entity position on the most recent proposal to reform the immigration process. As such, society members are invited to submit comments to Paul J. Kostek, 13517 Empire Way S#406H, Seattle, WA 98178, phone (206) 544-6816.

S.2104, the Immigration Act of 1988, was introduced by Senators Edward M. Kennedy (D-Mass.) and Alan K. Simpson (R-Wyo.). The bill includes a method of addressing potential immigrants with no family in the US by defining a weighted point system for qualification.

The proposed bill would create a new category of "selected immigrants." A person falls into this category of eligibility if he or she accumulates 50 or more points under a scale based on the criteria shown in Table 1.

The IEEE/US Manpower Committee has taken issue with several of the proposed reforms and is in the process of developing an IEEE entity position statement. Since no action is expected to be taken in the present session of the US Congress, the Manpower Committee is meanwhile attempting to gather support for the proposed IEEE entity position so it will be ready to participate when the legislation is reintroduced, possibly next year.

The Manpower Committee believes the proposed point system is not stringent enough and does not take into account the demand versus the supply of qualified labor. The proposed requirement can be achieved without the individual possessing any extraordi-

Table 1. Point system for selected immigrant qualification.

Criteria	Maximum points	Fraction of total
Age (10 points for 21-34, 5 for 34-45)	10	11%
High school diploma	10	11%
Bachelor's degree	10	11%
Graduate degree	5	5%
Occupational demand	20	21%
Occupational training or work experience	20	21%
English language skills	20	21%
Maximum achievable points	95	

nary skills.

The Manpower Committee recommends dropping the points awarded for a high school diploma and a bachelor's degree. The awarding of points should be based solely on advanced degrees, occupational supply and demand, occupational training or experience, English language skills, and age.

Another provision in the proposed bill is to permit flexibility in the present labor certification requirements. The Manpower Committee believes that this flexibility is not necessary. The present alien certification requirements not only ensure that immigrants receive a fair

opportunity to immigrate, but also assure that the supply of labor in the US is taken into consideration.

The Manpower Committee has used an existing IEEE Entity Position, "Alien Engineers, Foreign Students, and Our National Engineering Resource," from August 1983, as a basis for the currently proposed entity position statement.

A copy of the 1983 statement—as well as a summary of the proposed bill (S.2104)—may be obtained from the IEEE Washington Office, 1111 19th Street NW, Washington, DC 20036-3690.

IEEE membership dues to increase \$5 in 1989

For the first time since 1984, the IEEE Board of Directors has passed a \$5 dues hike to help support new and expanded membership services. The 1989 dues will go to \$57 from \$52.

Specifically, the dues increase will permit IEEE Spectrum to publish more pages; establish a transnational desk to help members in Region 8 (Europe, Middle East, Africa), Region 9 (Latin America), and Region 10 (Asia, Pacific Rim) overcome language, currency, and cultural difficulties linked to obtaining IEEE services; and defray the cost of

paper and postage for IEEE publications.

As reported on p. 78 of the August 1988 issue of *Computer*, the IEEE Computer Society's Board of Governors voted last June 17 to keep the amount IEEE members are levied for Computer Society's membership at \$15. Thus, 1989 will be the third straight year without a society fee increase.

The IEEE board has also voted to enable members to charge IEEE services on most major credit cards (minimum \$10).

17 campus computer science programs win accreditation

The Computing Sciences Accreditation Board has granted accreditation to 17 four-year US baccalaureate computer science programs during the current 1987-88 cycle.

Since it was established in 1984 by the IEEE Computer Society and the Association for Computing Machinery, the CSAB has accredited computer science programs at 65 public and private institutions. Twenty-three programs were listed during the first series of CSAB accreditation actions in 1986, and 25 were added in 1987.

The newly accredited programs are located at the University of Alabama at Huntsville, Appalachian State University, California State University at Fullerton, Howard University, Louisiana Tech University, New Mexico State University, the University of New Mexico, Oakland University, the State University of New York College at Plattsburgh, Polytechnic University in Brooklyn, New York, the University of San Francisco, the University of South Alabama, Southeast Massachusetts University, the University of Southern California, the University of Texas at El Paso, the University of Tulsa, and Virginia Commonwealth University.

All programs accredited by the CSAB must succeed in a rigorous year-long review process that begins with an in-depth self-study using materials specifically developed for computer science program evaluation.

Next, a team of distinguished computer scientists visits each campus for two days to verify the self-study and directly assess facilities, teaching effectiveness, and student work. Along with comments from the particular institution, a report by each visiting team is carefully weighed by the Computer Science Accreditation Commission made up of 30 computer scientists.

All CSAB officers and CSAC visiting team leaders and evaluators are volunteers from the Computer Society and ACM.

CSAB expects to evaluate another 35 programs during the 1988-89 accreditation cycle. To qualify for evaluation, a baccalaureate computer science program must be designed to prepare its graduates for professional employment and progressive careers as computer scientists.

Additional information can be obtained by contacting Karen Ellis, CSAB, 345 East 47th Street, New York, NY 10017, phone (212) 705-7314.

UPDATE

Contributions to Update are welcome. Send news of industrial or university research and of public policy or professional issues to Steve Wilcox, Computer, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720; Compmail +, s.wilcox; or to Bruce Shriver, Editor-in-Chief, Dept. of Decision Sciences, University of Hawaii, 2404 Maile Way, Honolulu, HI 96822.

East bowls over West in computer quiz

An East Coast-based team of computer industry experts came to Boston's World Trade Center, saw their West Coast competition, and conquered them handily in the first Computer Bowl October 7.

"It is a triumph of truth, justice and the American way—and minds filled with otherwise useless facts," said East Coast team captain Richard Shaffer, editor-publisher of the *Technologic Computer Letter*. The East Coast will join the Computer Museum in co-hosting the second Computer Bowl in 1990.

West Coast leader David Bunnell offered an alternate explanation for his team's loss: "People in the East study harder and read more. Out West, we're just too busy being successful and making money." Bunnell is chairman and CEO of PCW Communications.

"Since we knew we were smarter, we failed to take the contest seriously enough. While they crammed, we sat in our hot tubs," Bunnell said.

Questions included "Who was the first US President to use a word processor?" and "Which of these is not a computer: Maniac, Brainiac, or Illiac?" (The answers are Jimmy Carter and Brainiac, respectively.)

Shaffer's winning team boasted Esther Dyson, editor-publisher of the newsletter *Release 1.0*; David Hathaway, partner in Venrock Associates; Mitchell Kapor, chairman of On Tech-

nology and founder of Lotus Development; and John William (Bill) Poduska, Sr., chairman and CEO of Stellar Computer and founder of Prime and Apollo Computer.

Joining Bunnell from the West Coast were Adele Goldberg, president and CEO of ParcPlace Systems; William "Bill" Joy, cofounder of Sun Microsystems; Allen Michels, chairman and CEO of Ardent Computer and founder of Convergent Technologies; and Casey Powell, president and CEO of Sequent Computers.

William R. Hearst III, editor-publisher of the *San Francisco Examiner*, posed the questions. Mike Perkowski, associate publisher of *Computer Systems News*, judged the event. Christopher Morgan, former editor of *Byte* and *Lotus* magazines, served as emcee.

The PBS television show, *Computer Chronicles*, is scheduled to telecast the contest nationwide soon.

The event was sponsored by the Computer Museum to benefit the museum's educational programs and to help launch a national Junior Bowl program giving high school and college students a chance to test and add to their knowledge of computer history and information. The Computer Society is a corporate member of the Computer Museum, which is exclusively devoted to computers and their impact on society.

Report stresses ongoing education for engineers

American industry should upgrade its engineering work force through ongoing education, treating it as an intellectual resource that is as vital to international competition as technological and manufacturing advances, according to a new report from a committee of the National Academy of Engineering.

Focus on the Future: A National Action Plan for Career-Long Education

for Engineers calls for a nationwide coalition to "coordinate, monitor, urge, and advocate action for career-long education for engineers." The coalition would require participation from industry, academia, professional societies, government agencies, and private education providers.

In addition to its assertion that much of the knowledge gained through a formal engineering education is obsolete

within three to seven years, the report cites "forcing factors" that call the need for ongoing education into sharp focus, including global competition to upgrade engineering skills, rapid technological advances that quickly make knowledge obsolete, and projected reductions in the number of new engineers.

The report states that on-the-job engineering courses have worked well for the participating companies. Motorola experienced a "30-to-1 payoff in revenue" in one year, according to the report, and a combination of training and computer-aided design facilities at AT&T produced a five-fold increase in productivity over 10 years among that company's design and engineering staff.

"It would be unreasonable to expect that such large increases as those cited above could be attained consistently over a much expanded national effort," the report states, although it notes the committee's belief that the US engineering community is not close to realizing its potential.

The committee claims that investment in continuing education programs among smaller companies especially needs improvement. The report states that 65 percent of engineers at small, nonurban companies received no continuing education in a given year, while about 36 percent of engineers at large urban firms did not receive such instruction.

"Many companies can probably afford to invest more than the typical 2 percent of their engineering budget in education with profit," states a research paper contained in the report. Another paper included by the committee concludes that successful educational programs are those that are visibly supported by upper management, that involve company engineers and managers in the design and implementation, and that do not penalize participants for working hours spent on education.

The committee also suggests improvements in career education itself, including a greater variety of programs for working engineers, more incentives for outstanding teachers, and greater attention to such basics as convenient class times and locations and accessible support services, such as parking.

Funding for the study was provided by the National Science Foundation and the NAE's Technology Agenda Program.

Copies of the report are available from the Program Office, NAS 050, National Academy of Engineering, 2101 Constitution Ave. NW, Washington, DC 20418, phone (202) 334-1544.

Academy announces international engineering award

A gold medal and \$350,000 comprise a new international award announced by the National Academy of Engineering.

The first Charles Stark Draper Prize will be awarded in October 1989 to recognize achievements in engineering and technology "contributing to the advancement of human welfare and freedom." The award will subsequently be given biennially.

"This award is designed to focus worldwide attention on the critical role that engineering and technology play in improving the quality of everyday life," said NAE President Robert M. White in a press conference announcing the award. "Our society tends to reward the discoverer of basic scientific principles but overlook the engineer who puts that principle into practice in products and services that yield societal and economic benefits."

Recipients can be any living person or group, from any country and any engineering discipline, for a specific achievement or a body of work. Nominations will be solicited from members and foreign associates of the NAE, the National Academy of Sciences, the Institute of Medicine, and engineering academies worldwide; members of recognized US and international engineering societies; and other

individuals deemed eligible by the NAE.

The prize is endowed by the Charles Stark Draper Laboratory of Cambridge, Massachusetts. Draper developed the theory and invented and developed the technology of inertial guidance systems used in aircraft, submarines, missiles, and space vehicles. The systems use gyroscopes and accelerometers to determine a craft's position in space without any outside reference points such as stars, landmarks, or radio contacts.

The recipient will be selected by a committee appointed by the NAE and chaired by Robert C. Seamans Jr. of the Massachusetts Institute of Technology. Other committee members are Lew Allen Jr. of the Jet Propulsion Laboratory, Erich Bloch of the National Science Foundation, Harvey Brooks of Harvard, Solomon J. Buchsbaum of AT&T Bell Labs, Robert A. Chaprie of Cabot, Thomas E. Everhart of Caltech, John W. Fisher of Lehigh University, Allen E. Puckett of Hughes Aircraft, Simon Ramo of TRW, Daniel I. C. Wang and Sheila E. Widnall of MIT, and Alvin M. Weinberg of the Oak Ridge Associated Universities.

For more information, contact the National Academy of Engineering, 2101 Constitution Ave. NW, Washington DC 20418

Eliza creator to receive social-responsibility award

Joseph Weizenbaum of the Laboratory of Computer Science at the Massachusetts Institute of Technology will receive the 1988 Norbert Wiener Award for Professional and Social Responsibility from the Computer Professionals for Social Responsibility. The presentation will be made at the CPSR's annual banquet November 19 in Palo Alto, California.

A computer pioneer since the early 1950s, Weizenbaum developed Eliza, a computer program that simulates the dialogue between a Rogerian psychotherapist (the computer) and a patient (the user). The author objected when the program was pointed to as a potential form of actual psychotherapy. He detailed this and other objections to society's growing dependence on computers in *Computer Power and Human*

Reason: From Judgment to Calculation.

Weizenbaum has also worked as a long-time peace activist. In the late 1960s he was a founder and leader of Computer Scientists Against the ABM, a precursor to the CPSR. He is retiring from MIT this year.

The Norbert Wiener Award was established in 1987 to recognize individuals in the computing field or dealing with computing issues who make a significant personal sacrifice for or contribution to public safety, risk reduction, and standards of professional conduct. Wiener, an MIT mathematician and cyberneticist who died in 1964, was concerned about the responsible use of computers and opposed the militarization of science and technology.

Digital has it now.

The route to the 1990s.

Digital's Database Systems Groups are on the route to the 1990s. We're committed to moving forward towards future generations of Digital products.

Our winning organization is offering exceptional and diverse technical challenges to database engineers.

Move into the fast lane of database development with Digital in Nashua, New Hampshire or Colorado Springs, Colorado.

Database Product Development

Colorado Springs, Colorado & Nashua, New Hampshire
We develop internals of database management systems like Rdb/VMS, DBMS, Database Distributor and SQL. New areas of attention include extended relational models, knowledge-based systems and object-oriented database management systems. We are also developing the next generation of high performance, extended relational database systems using state-of-the-art technologies. These positions require an MS/BCS or equivalent and several years' experience in database internals, as well as expertise in:

- Distributed Database (distributed query optimization, concurrency control, distributed transaction management)
- Database Languages
- Multi-vendor Database Interoperability
- Relational Database Development
- Object-oriented Languages and Database Products

**DATABASE SUPERVISORS/PROJECT LEADERS
PRINCIPAL SOFTWARE ENGINEERS**

Database Performance Analysis

Colorado Springs, Colorado & Nashua, New Hampshire
We provide all predictive performance modeling support for all future database products, involving predictive analytical modeling as well as simulation modeling. We also conduct performance testing and benchmarking for current products.

MATH MODELER — Experience in analytic queuing network methodology and solution algorithms, and ability to map complex systems onto Markovian and/or queueing models.

SIMULATION ENGINEERS — Performance predictions by running event and/or process-driven simulations of designs.

PERFORMANCE TEST ENGINEERS — Run large-scale benchmarking lab, utilizing multi-processor, multi-systems configurations.

Database Architecture Colorado Springs

We are responsible for the industry's most advanced and versatile relational database architecture, including interfaces and languages design and compliance verification.

DATABASE ARCHITECTS — All openings require at least 3 years' applicable technical experience.

Move into the database development fast lane. Send your resume now, indicating your geographic preference to:

For openings in Nashua, NH:
Chris Larkin
Digital Equipment Corporation
Department 1201 8133
110 Spit Brook Road
Nashua, New Hampshire 03062

For openings in Colorado Springs, CO:
Mike Johnston
Digital Equipment Corporation
Department 1201 8133
1175 Chapel Hills Drive
Colorado Springs, Colorado 80920

We are an affirmative action employer.

digital™

NEW PRODUCT REVIEWS

Editor: Richard Eckhouse, MOCO, Inc., PO Box A, 91 Surtside Rd., Scituate, MA 02055, Compmail+, rackhouse

Exploring expert systems

Valerie Barr, Pratt Institute

Expert systems—also, and perhaps more properly, known by the term knowledge-based systems—seem to be everywhere. Various applications have appeared in corporate settings, including the well-known Xcon system designed by Digital Equipment and the Cooker system designed by Campbell Soup. Digital developed the former to automate the organization of computer system components and to ensure that order shipments included all necessary parts and cables. Campbell developed the latter when a 40-year employee who serviced the cooking machines approached retirement. The company incorporated his knowledge about the machines into the Cooker program to provide expert help to new service employees.

Various applications developed for home use largely serve as an introduction to knowledge-based systems. You can find these programs as samples in the various expert system shells, as shareware, and in ads in the back pages of popular computer magazines. The applications are often food-oriented—picking the proper cheese or wine to accompany a particular meal, for example.

Looking only at these two extremes can daunt computer users who believe they have an application suited for an expert system, but who are unclear on what expert systems actually are or how to build one. Expert system shells are programs designed to aid a user in building an expert system. The shell essentially provides a framework into which the user can plug the details of the application.

In this review, I take the perspective of a user developing an expert system for the first time. The shells reviewed fall in the under-\$500 price range, so the initial outlay is not prohibitive. For the most part, you do not need a knowledge of expert systems to use the shells, since many of them provide some introductory material as well as examples.

Introduction to expert systems

In expert systems, unlike other types of programs, we tell the computer what to know, not what to do. When constructing the expert system, we do not provide a set of instructions; rather, we provide knowledge and advice. If we have a step-by-step algorithm for solving the problem at hand, then a "traditional" program is in order. However, if we have no such step-by-step method, then an expert system is called for. Problems of this sort, for which we cannot write down a step-by-step solution, are considered ill-structured problems.

Ill-structured problems are rarely numeric, and you might find it impossible to write a set of specifications for them. Their solutions rely on a large body of knowledge or on things that cannot be quantified, such as 40 years of experience repairing cooking machines.

The AI approach to ill-structured problems looks at the ways people normally solve them. Following this approach, we want to accumulate knowledge and store it in such a way that we can then use it to solve future problems of the same type. In effect, we collect, formalize, and represent within a knowledge-based system some large body of knowledge specific to a task, then use that knowledge as if we had it in our own heads.

Knowledge-based systems have two basic parts: the inference engine and the knowledge base. The knowledge base contains the task-specific information. The inference engine is the mechanism for using the information in the knowledge base.

In a rule-based system, the knowledge base breaks down further into the rule base and the working memory (or database). The system usually (I would hope) has some kind of explanatory interface and a user interface. Also, some systems will have a knowledge

acquisition module.

A rule base is a set of rules specific to the problem at hand. For example, a knowledge base system for auto repair would have rules such as

IF	the engine does not start
AND	the starter motor does not turn over
THEN	the problem is in the electrical system
IF	the engine does not start
AND	the starter motor does turn over
THEN	the problem is in the fuel system

Each rule has two parts: the antecedent or premise (IF and AND parts) and the consequent or conclusion (THEN part). A rule is triggered if information in the database matches the antecedent of the rule. When the action specified by the rule is carried out, the rule fires and the conclusion of the rule joins the working memory.

Working memory, or the database, consists of a set of facts that describe the current situation. Generally, you will start out with very few facts. These will expand as you learn more about the situation at hand based on the rules executed.

The inference engine, also called the rule interpreter, has two tasks. First, it examines facts in working memory and rules in the rule base, and adds new facts to the database (memory) when possible. That is, it fires rules. Second, it determines in what order rules are scanned and fired. If the system is designed to ask the user for more information, then the inference engine does the asking, through the user interface. The inference engine is also responsible for telling the user what conclusions have been reached.

You use the knowledge acquisition module to add rules to the rule base and to modify existing rules. In a very simple system, you will have to use an editor or word processor outside of the

expert system. More sophisticated shells include an editor so that you have a mechanism for entering rule information.

The user interface is usually a natural language interface, which gives the user some degree of flexibility and a feeling of familiarity when interacting with the system. The user interface will provide some introductory message as well as ask questions of the user in a seemingly natural way.

The explanatory interface is that component of the system that lets the user see how the system reached a particular conclusion. Various degrees of sophistication are possible in the explanatory interface. For example, at any point the interface might allow the user to ask how the system reached an intermediate conclusion. Again, when the system asks the user for additional information, the interface might permit the user to ask the system to explain why it needs the particular information requested.

An expert system shell contains all or most of these components, except the knowledge base and the actual statements issued by the user interface. The person developing the system adds the rules, thus providing the knowledge base. The end user of the resulting expert system provides the information that goes into the database or allows information from the knowledge base to be placed into the database.

Knowledge-based systems employ two basic control strategies: backward chaining and forward chaining. To illustrate, assume a knowledge base containing the rules

```
R1: IF A AND B THEN C
R2: IF C THEN D
R3: IF E THEN B
```

and a database containing A and E.

Forward chaining involves searching the rules in the order listed to see which one can be fired based on the appearance of its antecedent in the database. R3 is the only one, since E is in the database. Fire R3 and add B to the database. Search the rules again. R1 can be fired because A and B are in the database. Add C to the database. Now R2 can be fired, and D is added to the database.

The forward-chaining inference engine algorithm summarizes this, as shown in Figure 1.

Backward chaining starts by trying to prove C, which is the consequent of the first rule. To prove C, we must have A and B in the database. A is already there. To get B, we look for a rule that has B as its consequent and try to estab-

lish the antecedent of that rule. A search finds that R3 has B as its consequent, with E as the antecedent. Since E is in the database, we can add B to the database. Now R1 can be fired, and C is proven or can be added to the database, or returned to the user as the result of the system. If the system was set up to allow multiple conclusions, it would proceed to R2. Since C was proven by R1, D could be proven by R2.

The backward-chaining inference engine algorithm summarizes this, as shown in Figure 2.

Applications that use forward chaining, such as process control, we call data-driven. Applications that use backward chaining we call goal-driven.

You should use forward chaining if you have a small set of relevant facts, where many facts lead to few conclusions. A forward chaining system must have all its data at the start, rather than asking the user for information as it goes.

You should use backward chaining for applications having a large set of facts, where one fact can lead to many conclusions. A backward-chaining system will ask for more information if needed to establish a goal.

The expert system shells reviewed below all implement a backward-chaining strategy, allowing you to design interactive knowledge-based systems, where the end user supplies the data.

In the interest of fair testing, I took an existing set of rules written with no particular shell in mind and set up systems with each shell using those rules. The following reviews describe the expert system shells ESIE, Exsys, and VP-Expert, and discuss my creation of expert systems using this given set of rules.

ESIE

ESIE, from Lightwave, stands for Expert System Inference Engine. This shareware package comes on one disk containing 19 files. It includes on-disk documentation consisting of a user's manual, tutor, knowledge engineer's manual, history (a brief history of AI), and a novice guide (for new users also unfamiliar with AI). ESIE is available directly from Lightwave or from shareware groups.

For the \$145 registration fee, you get a copy of the latest version of ESIE, printed versions of the manual, PC-Write, and access to the ESIE help line.

When a new fact is added to the database,

- (1) Scan the knowledge base to find all rules using that fact.
- (2) Discard those not ready to fire.
- (3) Add those ready to fire to the active list.
- (4) Fire the rule at the front of the active list.

Figure 1. The forward-chaining algorithm.

- (1) Find the list of rules that conclude about the current attribute A.
- (2) If there are no such rules, ask the user for the value of attribute A.
- (3) Otherwise, for each such rule R, use the following to determine the truth of the premise of the rule:
For each clause C in the premise of R:

Look in the database for the value of the attribute mentioned in C.

If there is no information in the database about that attribute

Start from the top with that attribute as the current one.

Evaluate the truth of C using the information in the database.

If the premise of the rule evaluates to "true," make the conclusion shown (given in the rule).

Figure 2. The backward-chaining algorithm.

If you register with Lightwave as a purchaser of ESIE, you will get the next version of ESIE when it is released.

System requirements are an MS-DOS computer with at least 128 Kbytes of RAM (256 Kbytes recommended), DOS 2.0, and a word processor that produces ASCII files.

Only three files are essential to run ESIE: Esie.com, Esie.cfg (created by the Config program), and the file containing the knowledge base. This last is built by the expert system developer in the word processor.

Once in ESIE, you have only four top-level commands: Go, which starts the consultation (meaning you run the expert system); Exit, which returns to DOS; Trace On, which helps you debug the knowledge base; and Trace Off, which turns the trace off.

The trace is off by default when you enter ESIE, and you must change its status before the consultation begins. If you are in the midst of the consultation and suspect a bug in the knowledge base, you cannot turn on the trace at that point. Instead, you have to finish the consultation, turn the trace on, and try again. A consultation will continue until it completes or an error is found in the knowledge base logic.

When you first turn the trace on, it tells you the number of rule lines (ESIE counts each IF, AND, and THEN separately, so it really does measure rule lines, not rules), the number of questions, and the number of legal answers in the knowledge base. Then, as the consultation proceeds, you see the subgoals or subproblems, that is, the information the inference engine is trying to find so that it can fire rules. The program displays new answers as they are determined, either directly from information you provide or from the rules.

The system has seven types of statements that you can put into a knowledge base. "Introtext is <text>" defines the message printed when the consultation starts. "Goal is <variable>" defines the variable for which the consultation is seeking a value. The rules are handled in a standard IF-AND-THEN form. In addition, the knowledge-base designer can specify legal answers, the text of questions asking the end user for information, text preceding the solution found by the system, and termination text signifying the end of the consultation.

With ESIE, it took some maneuvering to get my set of rules into the proper format. I have two suggestions that might help novice expert-system designers:

(1) Even if you have your application in mind, don't start writing rules until you know what shell you will use, because it can be harder to rework existing rules than to think them up in the proper form.

(2) I found it helpful to set up two windows in the word processor. In one, I put one of the sample knowledge bases. In the other, I wrote my own rules, following the general format given in the sample.

A knowledge base set up with ESIE can accept 1,000 rule lines (between 300 and 500 actual rules), 300 questions, 50 legal answers, 400 variables and values,

12,000 bytes of text, one goal, and one answer text. ESIE will look for up to 50 different rules to satisfy at a time (in attempting to prove a THEN), and it can learn up to 200 facts before it runs out of space in memory.

In setting up the rule base, you can put the rules and questions in any order. Since each question includes the variable about which it asks, you can have all the rules first, followed by all questions, or each rule followed by any

ESIE is a good system for someone who wants to try out expert systems or who has an application without complex rules.

related questions.

I did encounter a few frustrating points in using ESIE.

For one thing, you cannot edit rules from inside ESIE. When an error crops up, you have to exit ESIE, go to your word processor, and then return to ESIE after (you hope) fixing the error.

Another frustration, particularly when learning ESIE, is that you cannot load a new knowledge base from within ESIE. You have to return to DOS and reenter ESIE, specifying the new knowledge base.

In general, ESIE is a good system for someone who wants to try out expert systems or who has an application that, while it may involve a lot of rules, does not involve complex rules. It is certainly possible to develop very useful knowledge-based systems using ESIE.

Reader Service 21

Exsys (Version 3)

In comparison, Exsys from Exsys Inc. is a much more powerful system than ESIE. The first hint of this shows up in the system requirement of 320 Kbytes of RAM. The Exsys demo allows execution of knowledge bases with up to 80 rules, but you can save only 25 rules to disk. In the full system on a PC, you can run up to 5,000 rules. Exsys will use all available memory and requires about 64 Kbytes for every 700 rules (assuming an average of six or seven conditions per rule). On a VAX

computer, you will have essentially no limit on the number of rules.

Exsys costs \$395 for a single-user PC version, \$600 for a noncommercial runtime license for PCs (allowing unlimited copies), \$600 for a commercial runtime license for PCs (allowing 30 installations), and \$7,500 for a VAX version (allowing up to 16 users; \$12,500 for more than 16 users). The recently released Exsys Professional costs \$795 for a single-user PC version and \$12,000 for a 16-user VAX version.

The many features offered in Exsys include "why" and "what-if" capabilities. For the "why," Exsys shows the chain of rules used to arrive at a result. For the "what-if," the end user can change some of the answers given during a consultation and rerun the consultation. It is then possible to look at the old and new results together and compare them. You can also save input and results to disk.

Exsys allows the system designer to specify the probability that a certain result is true. You can set up an Exsys system in three different ways. First, a true/false (or 1/0) method gives all answers as yes or no. Second, the solutions use probability values between 0 and 10. Third, solutions give probability values from -100 to 100. When the program displays the results of the consultation, the end user can view all results or only those above a certain probability. Once you have chosen the probability mode for a system, you cannot change it.

The rules in Exsys can get more complicated. For example, you can carry out numeric comparisons in the IF portion and computations in the THEN part. Furthermore, the THEN part can have ANDs, so you can get multiple results based on the condition(s) satisfied in the IF portion of the rule. You can also have an ELSE section of a rule, so you can specify what results hold if the IF portion is not satisfied. Each part of a rule (IF, THEN, ELSE) can have up to 126 conditions. You can build OR conditions into the IF portion, but only with regard to the value of a single variable ("IF the color is blue or green" is okay, but "IF the color is blue or the time is night" is not okay).

Basically, the rules can ask only two types of questions: multiple choice or numeric value. In the first case, the user is asked a question and presented with choices, each preceded by a number. The user responds by entering the number(s) of the choice(s) selected. The response must fall within the range of numbers displayed or Exsys will not allow the user to proceed with the con-

sultation. In the second case, the user is asked a question (like "The length of the room is ___?") and gives a numeric answer.

When executing an Exsys-based system, you can call external programs for data acquisition, calculation, or result display, for example. You can also pass data back to Exsys for use in analysis during a consultation. This capability allows the designer to build systems with a great deal of power and flexibility.

Exsys' internal rule editor provides prompting, so you can often select material for a rule based on information supplied by prior rules. The editor takes a bit of getting used to, but after you enter about five rules, things start to go very quickly. Qualifiers receive numbers within the system when first introduced, so you can recall them for use in subsequent rules.

As designer, you also enter introductory and concluding text into the system, as well as your name as author and a name for the expert system. You also select the data derivation options (apply all possible rules or stop after the first successful rule) and the default for rule display. If rule display is on, then each rule used will be shown when it completes firing, with either the IF and THEN sections or just the ELSE section highlighted, depending on whether or not the condition in the IF section was satisfied. Regardless of the default you choose, a person using the expert system can override it.

During system design, you can add additional information to the rules in the form of notes and references. A note is for the user's information and is displayed with the rule if rule display is on. A reference can explain the source for the rule. This information is displayed only if the user asks for it.

I encountered a few bothersome things about the demo disk. First of all, the file allocation table was faulty, causing the demo of the report generator to bomb. Furthermore, while the demo is very good and can certainly teach you enough to start building a system, there is no way to move through it quickly. So, if you want to repeat parts of the demo to refresh your memory, you have to restart from the beginning. The demo also does not tell you how to quit in the middle (Ctrl-Break works).

Fortunately, the Exsys manual does not refer to the antecedent portion of a rule (the IF part). Instead, it refers to a "condition" with two distinct parts: the qualifier, which contains a "verb" (for example, "the color is"), and "values," which are possible comple-

tions of the condition. While it doesn't take long to get used to this, it can confuse you at the start.

The demo's major flaw is that it does not clearly explain that, if you have a rule in which the THEN section sets a qualifier, then you must have an ELSE section that sets the qualifier to some other value. Otherwise, the system will ask the user for it.

In executing a system, Exsys tries to be efficient where space allows. When you create a knowledge base, Exsys builds two files for it, a .rul and a .txt file. When a consultation begins, Exsys reads in the .rul file, which contains an encoding of the rules. Then, if there is space, Exsys also reads into memory the .txt file, which contains the actual text of the rules. This allows the consultation to proceed more quickly. If the .txt file does not fit in memory, the program will access it from disk when needed, slowing down the process.

Overall, Exsys has a myriad of features that allow you to design very powerful expert systems in a fairly efficient manner. In addition to the features discussed here, it has a report generator that lets you design systems that output to files (or the printer) various information generated by the expert system. It can also do a mix of forward chaining and backward chaining or pure forward chaining. Use of these and other advanced capabilities involves some study of the manual, particularly the section on command-line options.

Clearly, Exsys is designed for the development of sophisticated and complex knowledge-based systems, but it also works well for small-scale systems. I think it is a good choice for a user who plans to ultimately design large systems, but would like to start out with smaller scale systems.

Reader Service 22

VP-Expert (Version 1.2)

Of the three packages, VP-Expert by Paperback Software is the one most representative of the current generation of PC software, both in look and feel as well as capabilities. It is on a par with Exsys in terms of system configuration, requiring 384 Kbytes of RAM and DOS 2.0 or higher. While you can probably build knowledge bases as large as the ones built with Exsys, what sets this shell apart is its use of the PC and its compatibility with other PC software packages. It costs \$100.

The program source for VP-Expert is in Microsoft C, allowing you to build knowledge bases that use all of primary

memory (not extended memory). The program has a limit of 20 conditions in a rule and 20 levels of backward chaining. VP-Expert also has a knowledge base chaining feature, allowing you as designer to build knowledge bases which otherwise would not fit into memory. At the end of one knowledge base, the program saves the current database values. A Chain command then opens a new knowledge base. Finally, the database values are restored.

VP-Expert does not have an on-disk demo or tutor. However, a number of sample knowledge bases come as part of the package. These are referred to in the first chapter of the manual, which serves as a tutorial. The screen displays shown in the tutorial chapter are very accurate. The only discrepancy is that the book consistently states that the light bar will come up on option 1 (Help), but it consistently comes up on option 2 instead (which varies from screen to screen). After the first chapter, I noticed slight differences between the screen displays pictured in the book and the ones that came up on the computer, probably due to changes in the sample knowledge bases.

Each screen in VP-Expert comes up with command choices clearly displayed along the bottom of the screen. You can select one of these choices by typing the first letter, typing the command number, hitting the function key corresponding to the command number (F1 for command 1, etc.), or using the cursor control keys to move the light bar to the desired command and pressing Enter. However, when you use the built-in editor, only the latter two methods work. The others cause unwanted characters in your file. I suggest choosing a single method early, one you can use for everything. Otherwise, the editor will take you by surprise when you start using it.

The VP-Expert main menu presents six options (besides Help and Quit). FileName and Path let you call a new knowledge base or change the default drive and directory. A novice will see the options Consult and Edit most frequently. Selecting Consult takes you to a consultation window, where an expert system executes. During development, the consultation window takes up half the screen; a rules window and a results window occupy the lower half of the screen. The rules window shows the activity of the inference engine, while the results window notes conclusions derived during the consultation. In a finished system, ready for the end user, you can remove the extra windows by entering a single statement into the

knowledge base to get a consultation window the size of the entire screen.

While in the consultation window, when the consultation has ended or paused (waiting for user input), you can obtain a subsidiary menu by typing a slash (/). This menu gives choices for How (how the value of a variable was found), Why (why the current question is being asked), and Slow and Fast, which slow down or reset the speed of knowledge base execution (so that you can read the display in the rules and results windows).

Other options available from the consultation window are

- **WhatIf**—lets the user change a single response and rerun the previous consultation. All other variables keep their prior value unless changed by rules in the rule base. The user is asked for new information if needed to satisfy some rule which had not fired during the initial consultation.
- **Variable**—lets the user select any variable to see the value it had at the end of the previous consultation.
- **Rule**—lets the user specify a rule number to see that rule displayed in the rule window. If no rule window is open, nothing happens.
- **Set**—displays a submenu that has commands for Trace, Slow, and Fast. Selecting Trace causes the path of the inference engine during the next consultation to be stored in a file with the

same prefix as the knowledge base and a `trc` extension. The user can view this later. Slow and Fast are the same as above. The Slow command has a cumulative effect if issued several times in succession.

Selecting Edit from the main menu puts you into VP-Expert's built-in editor. An excellent editor for this purpose, it has relatively few quirks. When you enter the editor, the function key assignments appear at the bottom of the screen. If you press Ctrl, Alt, or Shift, the display changes to show the assignments for the function keys in combination with the depressed key (Ctrl-F1, Ctrl-F2, etc.). One apparently undocumented feature in the editor can give you quite a surprise—Ctrl-Backspace deletes the word before the cursor.

A very nice feature of this package is the connection between the Edit and Consult modes. If a problem with the knowledge base materializes during a consultation, then VP-Expert returns to Edit mode at the point in the knowledge base where the problem was detected.

Knowledge bases designed with VP-Expert have a basic layout of actions, rules, and statements. The actions block provides overall direction for the consultation, displaying an introductory message, defining the goal variable for which the system is to find a value or values, and displaying a final message with the result. The actions block can

also include clauses specifying more complex tasks involving databases and spreadsheets.

The rules used are if-then rules, broken down into name, premise, and conclusion. Each rule is introduced by the keyword "rule" followed by a unique rule label, a string up to 40 characters long. The premise, or IF portion, can include up to 10 conditions. Each condition compares a variable to a value, using standard relational operators. You can also write conditions that compare a variable to the current value of another variable.

Conditions are combined using AND and OR but, unlike most other uses, OR takes precedence over AND. One problem with the introductory material is that the first mention of AND and OR does not explain the precedence.

The conclusion, or THEN portion, must contain at least one equation assigning a value to a variable. Conclusions can have confidence factors with them, using an integer between 0 and 100 to indicate the degree of certainty that the conclusion is valid. (Confidence factors can also be entered by the user at question prompts, reflecting the user's degree of confidence in the response. The system assumes a factor of 100 if you give no factor; a factor of less than 50 is treated as a negative response.) Multiple conclusions can be listed and do not use an AND for conclusion.

Comparison of sample rules from ESIE and VP-Expert

The expert systems set up for this example are designed to tell you what kind of drink to serve, based on information that you—as the end user—provide about your dinner guests and the entree you have chosen to serve. The rules are shown in the format used for each shell.

Sample rules from ESIE

```
IF expensive_wine Is yes
And New-Year's Is yes
Then type.drink Is Bond's-Champagne

IF expensive_wine is yes
And steak is yes
Then type.drink Is Chateau-Earl-of-Bartonville-Red

IF cheap_wine is yes
And chicken is yes
And well-liked Is no
Then type.drink Is Honest-Henry's-Apple-Wine
```

Sample rules from VP-Expert

```
RULE 1
IF Expensive_Wine = yes AND
New_Year's_Eve = yes
THEN Drink=Bonds_Champagne;

RULE 2
IF Expensive_Wine = yes AND
Entree=steak
THEN Drink=Chateau_Earl_ofBartonville_Red;

RULE 3
IF Cheap_wine = yes AND
Entree=chicken AND
Well_liked = no
THEN Drink=Honest_Henry's_Apple_Wine;
```

The statements section contains information about the consultation itself. In this section you enter the Ask command for those variables whose values the end user will supply. The Plural command indicates that a variable can have more than one value at a time during a consultation.

The Runtime command suppresses the rules and results windows during consultation. You can use other commands to change the background color of the consultation screen or to provide the end user with a menu of choices when asked the value of a variable.

VP-Expert contains on the order of 50 keywords, predominantly statements or clauses. The clauses are used in the actions block or in the conclusion of a rule. In the actions block, they are executed sequentially. However, clauses in a rule conclusion are executed only if the premise of the rule is true. The clauses let you display text on the screen, eject the printer, clear the screen, specify that the value of a variable should be found, and link VP-Expert to database and spreadsheet files.

VP-Expert's implementation of backward chaining is as follows: When the inference engine needs the value of a variable, it looks for a rule with that variable in the conclusion and attempts to prove the premise of the rule. If it finds no such rule, then it looks for an Ask statement for the variable to ask the user for information. If it finds an Ask, it will also look for a Choices statement for the same variable, which causes display of a menu with the options possible for the variable. If it finds no Choices statement, then it asks the user for the value and the user types a response.

Other important main-menu choices are Induce and Tree. Induce generates a knowledge base from an induction table. You can build this table in any ASCII text-editor, or you can use database or spreadsheet files (from VP-Info, dBase, VP-Planner, Lotus 1-2-3, and dBase and Lotus work-alikes). Each line of the table becomes a rule of the knowledge base.

Even when setting up your first simple system, it can help to build a table and use Induce. VP-Expert has strict rules about punctuation, and using Induce results in a model that you can then follow when adding additional rules to the knowledge base. Although the induction process is not very complex, it would be helpful to have more than one example of induction tables in the manual.

The Tree menu lets you view the path previously saved in a trc file. You get

options for text or graphics, the latter requiring a graphics card. The text display has three types of lines: <variable name>, which indicates that the value of the variable is currently being sought by the expert system; "Testing #," which indicates that the rule with the specified number is being tried; and (text), which means that the value in parentheses has been assigned to the variable being sought. Levels are indicated by indentation marked with ! to show the tabs.

The graphics display shows a tree construction with the root at the left-hand side of the screen. You can then zoom in to areas of the tree to see the detailed information, which is the same as that displayed in the text version.

The one complaint I had about the Tree menu is that it is the only menu that does not tell you how to exit and return to the main menu.

Overall, VP-Expert is an excellent package, particularly well suited to people familiar with light-bar menus and other features common in PC packages. The accompanying text is excellent as well, including, in addition to the beginner's guide (Chapters 1-4), chapters on database access and worksheet access, a topical reference, a menu command reference, an alphabetical keyword reference, a math function reference, and several helpful appendices. The index is also fairly good. One weakness is that it does not often reference the first four chapters, even though many topics appear there first. So, if you want to go back and review that material, often you must flip through those sections until you find the appropriate

part.

A final note: Paperback Software is now shipping Version 2, for \$249. They have made a number of substantial changes, including: forward chaining and "whenever" rules that fire whenever the condition in the premise is satisfied, even if backward chaining is being used (to have pure backward chaining, do not use this kind of rule in the rule base); hypertext with mouse support; dynamic images that allow display of graphical images (like a thermometer) that are tied to a variable so, when the variable changes, the image changes, and vice versa; automated access to dBase files; more powerful access to external programs; and smart forms ability.

Even at the new price, VP-Expert is an extremely powerful product for the money and at least as powerful as packages in higher price ranges.

Reader Service 23

Recommendations

If you have an application that seems suitable for an expert system or if you just want to see what this new technology is all about, you will find a number of options on the market. The three packages reviewed here suit novice users while providing varying degrees of growth potential.

Next comes a review of expert system shells by Peter Raeth, covering shells from several different price ranges.

Two PC-based expert system shells for the first-time developer

Peter G. Raeth, Air Force Systems Command

Have you ever wanted to try out a new software technology but run into the classical Catch-22? You know your company needs new technology, but the company wants proof of its cost effectiveness before spending much money. Yet you can't generate the proof because the seed money is just not available.

By using a few "nickels and dimes" scraped up here and there, software developers can begin exploring new software approaches using technologies already in place, such as microcomputers. This personal effort can gener-

ate the proof required for larger expenditures.

An increasingly popular software technology well worth exploring is expert systems. A fundamental difference distinguishes an expert system shell from a traditional programming language: shells are object oriented and knowledge intensive, while traditional programming languages are procedure oriented and code intensive. This fundamental difference affects even the way you have to think about an automation problem.

Shells can greatly simplify certain

automation problems, particularly those requiring great flexibility and incremental additions of capability. Using a natural mechanism, shells permit the capture of knowledge in the expert's own language. The key is that the shell doesn't change just because the knowledge base changes. Traditional programming techniques require that all the expertise be embedded in the code. The code has to change every time the knowledge represented by the code changes. Shells permit the developer to concentrate on the expert's knowledge instead of computer code.

Expert systems offer many implementation techniques. A frequently used technique that focuses on object-action pairs requires specific actions to be performed when specific objects exist. (However, objects do not necessarily exist in any predetermined order.) Usually, we think of these object-action pairs as if-then rules. If-then rules state, "If a given object exists, then perform a given action."

Choosing that first shell can be difficult because of all the various features. The accompanying recommended reading list contains articles that offer a good discussion on features for new users. Besides features, the new user should look for a shell at a reasonable price that he or she can use to develop meaningful applications and later expand to accommodate even larger applications.

Two PC-based expert system shells on the market support the idea that the cost of a tool should not exceed its value to the user and that the tool should grow with the user's needs. Clips, for C Language Production System, was written by NASA and is available on the user-supported market. Personal Consultant from Texas Instruments is a commercial product available as a series of additive modules. Both Clips and Personal Consultant come with a tutorial guide, a reference manual, example knowledge bases, and the possibility of on- or off-site training. In their most basic form, both shells work on the IBM PC and PC clones. For larger applications, they both require PC-AT class machines. However, even their most basic forms permit the implementation of useful expert systems.

I discuss these two shells from the viewpoint of the developer exploring expert systems technology for the first time. The basic question I'll address is, "Given a new developer and a detailed specification, did the shell prove effective?" To answer this question, I'll begin with an overview of an application's specifications. Next, I'll look at each shell's performance in meeting the

Recommended reading

- Barr, Avron, Edward A. Feigenbaum, and Paul R. Cohen, *The Handbook of Artificial Intelligence*, Vols. I, II, and III, Addison-Wesley, Reading, Mass., 1981 (Vol. I) and 1982 (Vols. II and III).
- Citrenbaum, Ronald, James R. Geissman, and Ronald Schultz, "Selecting a Shell," *AI Expert*, Sept. 1987, pp. 30-39.
- Forgy, Charles L., "Rete: A Fast Algorithm for the Many-Pattern/Many-Object Pattern Match Problem," *Artificial Intelligence*, Jan. 1982, pp. 17-37.
- Freedman, Roy, "27-Product Wrap-Up: Evaluating Shells," *AI Expert*, Sept. 1987, pp. 64-74.
- Gevarter, William B., "An Overview of Expert Systems," PB83-217562, Nat'l Tech. Info. Service, Franconia, Va., May 1982.
- Gevarter, William B., "The Nature and Evaluation of Commercial Expert System Building Tools," *Computer*, Vol. 20, No. 5, May 1987, pp. 24-41.
- Jackson, Peter, *Introduction to Expert Systems*, Addison-Wesley, Reading, Mass., 1986.
- Naylor, Chris, *Build Your Own Expert System*, Halsted Press, New York, N.Y., 1985.
- Krutch, John, *Experiments in Artificial Intelligence for Small Computers*, Howard W. Sams, Indianapolis, Ind., 1981.
- Raeth, Peter G., and James M. Hardin, "Using Expert Systems Technology to Develop a Statistical Strategist," Paper presented at the 51st Meeting of the Mississippi Academy of Science, 1987. Available from Raeth at AFWAL/AAWP-1, WPAFB, OH 45433.

specifications. As you'll learn, each shell stands on its own as a useful tool that meets the goals of expandability and value.

The application

For this review I implemented the Statistical Strategist, which can—given the assumptions attendant on a given data set and the purpose of a given analysis—recommend appropriate methodologies. This application was originally written in Turbo Pascal for the IBM PC, using expert systems techniques. It is supported by PC-File, a relational database management system, and PC-Write, a word processor.

Figure 1 shows a diagram of the application's logical structure.

The following paragraphs list major features of the Statistical Strategist application.

- User help facility. The user can ask for an explanation of words, phrases, and suggested methodologies without interrupting the flow of interaction with the system.
- Easy knowledge base entry. The domain expert, a statistician in this case, can easily enter and change methodologies and their related assumptions and purposes. A simple frame-based entry method captures this information in a relational database. You enter explanations into a word processor. You can enter both types of information in a random manner. The statistician does not have to worry about rules, information ordering, or special syntax in the development of the knowledge base.
- Interactive conversation that makes no initial assumptions. The Statistical Strategist makes no assumptions about which methodology will be appropriate. It queries the user to find out what it needs to know and makes recommendations as it acquires sufficient information. Its conversation is very direct, guiding the user quickly to an applicable methodology. It also explores methodologies that require more detailed assumptions.
- Machine translation of the knowledge base. There is no human interaction in the process of translating the knowledge base into something the inference engine can easily deal with. Once this translation is complete, the computer can initiate an interactive conversation with the user to determine the purpose of the analysis and the assumptions made on the data. As this conversation progresses, the computer recommends appropriate statistical methodologies.
- Logic trace facility. At any point in the conversation, you can ask the computer to show the reasoning so far. In this way, you can see the facts used by the computer to make its recommendations. At the end of the conversation, you can see which assumptions and purposes resulted in which methodologies.
- Restart capability. You can tell the computer to ignore the previous conversation and start the session again. This permits the user to change the answers to some questions, resulting in new questions being asked.
- No time delay between questions and recommendations. No delay occurs between the time the user answers a question and the time the inference engine delivers the next question or the

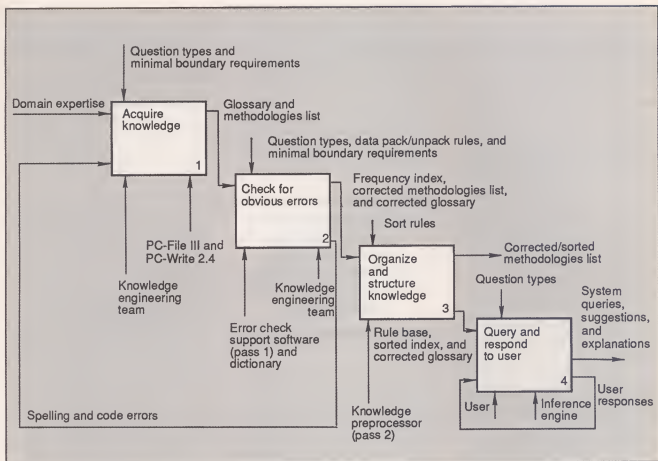


Figure 1. Logical structure of the Statistical Strategist.

next appropriate recommendation.

- Debugging capability. A knowledge base preprocessor helps the user catch entry errors, misspellings, and nonfactual entries.

Now let's see how Clips and Personal Consultant performed during implementation.

Clips (Version 4.11)

According to Clips' documentation, the Artificial Intelligence Section at NASA's Johnson Space Center has encountered a number of problems delivering Lisp-based expert systems to NASA users. Three problems in particular have hindered the use of expert systems within NASA: low availability of Lisp on conventional computers, high cost of current technology Lisp tools and hardware, and poor integration of Lisp with other languages, making

embedded applications difficult. Clips was developed to solve these problems.

Clips is command driven. Its primary representation methodology is a forward-chaining rule language with an inferencing technique based on the Rete algorithm. The basic elements of Clips are a fact list that forms a global memory for data, a knowledge base that contains all the rules and initial conditions, an inference engine that controls execution and decides the rules to be executed based on the available facts, and external software written by the user in C or some other language. C language code can be linked directly into the tool.

Getting started. I recommend starting with the *User's Guide*. Written in a conversational tutorial style, this guide is essential to the effective use of Clips' capabilities. If you are new to expert system shells, expect to spend at least two hours per day for two weeks reading the guide and working through two

meaningful exercises per chapter. The exercises are very important because they teach not only how to use Clips, but also good rule-programming style. Indeed, without good style, you cannot successfully complete the later exercises. Moreover, good style will keep you out of many traps as you go on to develop that first real application.

One thing missing from the tutorial is an installation guide. You need this supplement to specify the hardware and operating system expected by the given version of the shell. The guide should also explain how to set up the software.

Without an installation guide, I went down many a wrong path. My version of Clips runs on an IBM-PC. It nominally needs 230 Kbytes of RAM and PC-DOS version 3.x. It also turns out that Command.com must be on the boot-drive disk's root directory. Not knowing that last fact caused me no end of frustration. Clips will report that certain commands executed correctly, even

if they don't due to Command.com's absence.

To be fair, the disk label does say that MS-DOS 2.11 is required. MS-DOS typically indicates a need for an IBM PC or its clones. In fact, the NASA-compiled code even worked on a Zenith Z100, a computer that normally is not software compatible with the IBM PC.

Even with the problems caused by a missing installation guide, I must laud NASA for its well-written *User's Guide* and the companion *Programmer's Reference*. The *Programmer's Reference* tells you how to compile the Clips source code (written in C), how to add features, and how to take features out. It also explains Clips in complete technical detail. Both guides are well indexed, easy to use, and easy to understand.

Another great point is the user help desk provided by Computer Sciences Corp. (under contract to NASA, for government users only). Before I told them I was writing an article, I called with a question I knew they would not be able to answer readily. They took my name and number and called me back later the same day with a well-thought-out (and correct) answer.

A final note about Computer Sciences: They offer on- and off-site training. They can also fill you in on Space Operations Automation and Robotics (SOAR) conferences where Clips tutorials are given. I have not attended one of those tutorials, but I have a copy of the handouts. If those who teach the tutorial are as knowledgeable as the ones who run the help desk, the SOAR tutorial would be worth going to—once you have worked through the *User's Guide*.

Clips' syntax. Clips' syntax is very similar to Inference Corp.'s ART (Automated Reasoning Tool) and takes several lessons from Lisp. Even those with some experience in Lisp will find Clips' syntax and case sensitivity unnatural at first. However, with a little experience, you will see that the syntax does make a lot of sense and that it permits a more powerful and flexible system, especially when you consider that even Clips' user commands can be activated by rules. So, have patience with the syntax.

Implementing an application using Clips. The following paragraphs detail the major features of the Statistical Strategist from the Clips point of view.

User help facility. A user help facility was simplicity itself to implement. I

could have taken two approaches. One approach uses autotranslator techniques to generate Clips rules containing the help information. The other approach leaves the help information in a PC-Write data file and accesses it via a program external to Clips. I took the second approach, since that saved Clips' on-line memory. When the user types in "help," the help rule executes the System command. This command activates the external program, which asks the

In Clips, the developer must translate the domain expert's knowledge into rules and facts while following a special syntax.

user what help is desired and then displays the appropriate off-line information. Granted, this is slower than having the information on-line directly in Clips. This is one of the places where, as developer, you could modify Clips itself. You could translate the code used in the external program to C and integrate it into the Clips software. That would speed things up considerably while keeping all the text information on disk.

The method I chose to add the help capability to Clips took about five minutes to implement, thus showing one of the real powers of Clips. Like the original Statistical Strategist, written in Pascal, the Clips-Rules version was built incrementally. I found it much easier to add capability to the Clips version than to the Pascal version because of Clips' object orientation versus Pascal's procedural orientation.

The type of application I used seems more natural in the object-oriented setting. Of course, there are times when, even in this setting, you must write some procedural code. Clips makes it possible to have the best of both worlds. I found that Clips let me concentrate on what I wanted to do rather than on the details of how to do it.

Easy knowledge base entry. There are two ways to develop a Clips knowledge base. You can enter rules and facts directly into the system, test them, and then save them to disk. Alternatively, you could use a text editor (Clips has one built in), type in Clips rules and facts, and then use the Load command

to bring the result into the system. You can then perform tests, enter new rules and facts, and save the result to disk.

The commands *Defrule*, *Deffacts*, *Assert*, *Reset*, and *Run* are the major commands used for building and testing an expert system in Clips. However, the system has a host of other commands and capabilities. All of these made the implementation of the Statistical Strategist a fairly straightforward task, once I discovered how to install Clips and gained a bit of experience.

The problem with Clips' method of knowledge-base entry is that the developer must translate the domain expert's knowledge into rules and facts while following a special syntax. The original Statistical Strategist permitted the domain expert to enter knowledge into a simple frame, with the computer doing the translation. So, I used the original frame-based entry method implemented in PC-File and wrote a two-page program in Pascal that reads that database and translates it into Clips rules and facts. Then I wrote a header of about 10 rules to implement some standard features that would not change when the knowledge base changed. The original knowledge base translated into about 30 Clips rules. When I loaded the header and autogenerated knowledge bases into Clips, the resulting system worked much like the original.

Interactive conversation that makes no initial assumptions. This feature followed naturally from the autogenerated knowledge base and the handwritten header. It was not difficult to add more and more conversational power to the system. I just had to extend the autogenerator's rule outlines to offer more information and feedback.

Machine translation of the knowledge base. You must enter Clips knowledge bases directly in a special rule/fact syntax. No higher level entry method is provided. Of course, you could always extend Clips to do this, since it comes with the source code.

Logic trace facility. Again, I found this an easy task to implement. As the rules execute, the system asserts facts that Clips ignores but still stores in the order created. These facts contain the assumptions and purposes indicated by the user and the resulting methodologies suggested by Clips. When the user asks "why," the Why rule executes the Facts command. At that command, Clips lists the facts previously asserted. Due to continuous facts cleanup, the list contains no extraneous facts. Only the

trace of the conversation appears in the list. Once the previously asserted facts are listed, Clips picks up the conversation where it left off.

Restart capability. Clips easily empties its store of facts and begins executing anew. I only had to have a restart rule to execute the Reset command. Restart can be initiated at any time by the user or automatically by Clips, once there is nothing else for Clips to ask or recommend.

No time delay between questions and recommendations. On a minimally configured IBM PC, Clips does exhibit a slight delay between user responses and the next question or recommendation. These delays result from the Rete algorithm's attempts to find the next set of appropriate rules based on any new facts. This delay is less than one second.

I didn't notice a significant increase in the delay even with 342 rules and five initial facts loaded. The delay did increase somewhat, to about five seconds between the first response and the second question, in a test with 250 rules and 150 initial facts. This time decreased as the conversation progressed.

Clips took about 1 minute 23 seconds to load the Statistical Strategist knowledge base and begin the conversation.

Debugging capability. Clips' Watch command permits the developer to observe facts being asserted and retracted. You can also see rules executed.

Clips provides complete syntax checking. It clearly shows the error and its location. The *Programmers Reference* provides complete details on each error message.

The system makes no attempt to verify entries to the knowledge base. Clips assumes that, once the syntax is correct, the rule or fact is correct.

Clips in summary. Clips could not directly implement all of the original Statistical Strategist without additional programming in C or some other language. However, you must realize that Clips is a general purpose, rule-based shell. As such, it works well and is worth the price. Since it comes with the source code, Clips can be expanded to deliver additional capability. It can also execute external programs written in any language.

Be aware that Clips is not a commercial product. Thus, it does not have all the support some people—particularly nongovernment users—are used to. So far, NASA appears to have a solid com-

mitment to its tool. This should continue as long as there is a strong government audience. Given that ART's syntax is so similar to Clips (according to NASA), it should be possible to upgrade to this commercial tool if necessary.

As a government-produced, user-supported program, Clips is not copyrighted. You can freely distribute both the compiled and source code. Thus, you only need to buy Clips if you

TI's Personal Consultant modules support the beginner while giving the experienced knowledge engineer extensive power.

don't know someone who already has it.

Clips runs on the DEC VAX 11/780 under VMS, Sun workstation under Unix, Apple Macintosh, and IBM PC and AT or clones under PC-DOS 3.x and MS-DOS 2.11.

Since C is supposedly a very portable language, you should be able to compile and run Clips if you can get the source code into your machine.

Contact information. Government users send six diskettes or a tape of the type used by the intended computer. Contact Steven Baudenist, Computer Sciences Corp., Clips Users Help Desk/M30, 16511 Space Center Blvd., Houston, TX 77058, phone (713) 280-2233 or (713) 280-2430.

Nongovernment users pay \$250. Specify MSC-21208 and be sure to say what media you want. Contact Director, Software Distribution, COSMIC, University of Georgia, 382 E. Broad St., Athens, GA 30602, phone (404) 542-3265.

ART: Reader Service 24

Personal Consultant Plus (Version 3.02)

Texas Instruments has developed a choice set of integrated tools for the knowledge engineer, presented as a group of upwardly compatible modules called the Personal Consultant series.

These modules permit a new developer to get started without a lot of pain and to gradually move on to the development of very sophisticated expert systems. They support the beginner while giving the experienced knowledge engineer extensive power.

TI's modules include PC Easy, the foundation module, for \$495; PC Plus, an extended form of PC Easy, for \$2,950; PC Scheme, an integrated Lisp, for \$95; PC Images, a graphics-oriented user interface, for \$495; and PC Online, a real-time inferencing package, for \$995. If you purchase PC Plus within six months of PC Easy, the price drops by \$495. PC Scheme comes with PC Plus, but also as a stand-alone package. PC Images and PC Online are expander packages for PC Plus. Figure 2 shows how these modules are integrated.

Personal Consultant is menu driven. A rule language forms the basis for this expert system shell. While it can do forward chaining, its most natural inferencing style is backward chaining. For each goal conclusion, it goes sequentially through the goal's list of premises and thereby seeks to prove the conclusion. Each goal premise that cannot be proven directly becomes a subgoal. The rules that affect the subgoal are executed in an attempt to prove the subgoal. Premises of subgoal rules that cannot be proven directly result in other subgoals. This process continues until the original goals are either proven, disproven, or become undeterminable.

Personal Consultant's inferencing method is supported by several major categories of information:

- rules: IF-THEN premise-conclusion pairs
- meta rules: rules affecting the execution of rules
- frames: groups of rules that can inherit attributes
- premises: what must be proven during inferencing
- goals: the results of the inferencing process
- parameters: objects that make up premises and goals
- variables: names of values accessible to the application
- functions: developer-defined Scheme modules
- text tags: blocks of text accessible to the application
- confidence factors: certainty of premise and goal values
- global properties: information defining the system's environment
- external software: user-written in Scheme or another language
- external data: sequential and dBase files

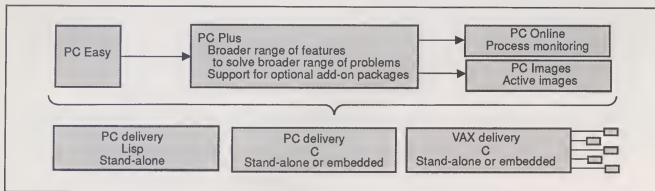


Figure 2. The integrated modules of the Personal Consultant series.

- still graphics: explanatory screen frames stored on disk
- moving graphics: screen images that move to assist user input

Getting started. Normally, I would advise you to start with the user's guide. However, the Personal Consultant series comes with more than nine manuals.

You should not just dive into this package without some prior thought. Start with the flyer "The TI Personal Consultant Series," published by the Texas Instruments marketing department. It gives a good technical and functional overview of the modules and how they are integrated. This short pamphlet will help you decide on the module(s) you need and the minimal hardware necessary to run them.

Pay special attention to the hardware requirements. For instance, do not purchase something that will restrict you to PC Easy or CGA graphics. On the memory requirements, I recommend 640 Kbytes of main memory even for PC Easy. (The Statistical Scientist is about all PC Easy can handle on a minimally configured IBM PC.) You will need at least 2 Mbytes of extended memory to achieve useful speed with the other modules. TI's software will work in less memory, but garbage collection and disk accesses for overlays slow things down considerably.

Table 1 summarizes the software options and the minimum hardware requirements of the Personal Consultant series of modules. I carried out this review of Personal Consultant on a fully configured Zenith Z248 using Personal Consultant Plus.

Once you have gotten an overview of the modules, you are ready to look at the manuals. Basically, each module comes with two manuals, a user's guide written to teach you how to use the module and a technical reference to

answer detailed questions. Each manual has two tracks though it, one track for the person new to expert systems and a second track for experienced knowledge engineers learning Personal Consultant's syntax and conventions.

If you are new to expert system shells, expect to devote two to three hours per day for five days with each manual. Be sure to work through the examples. Start with PC Easy's user's guide and explore your first application with that module before you purchase any of the other modules.

The sample expert systems sent with the Personal Consultant Series are discussed in the user's guide. They do a good job of showing the power of the shell and specialized tools like graphics screens. My complaint about the still graphics frames is that TI does not give you a lot of help to put them together. The manuals make a vague reference to a third-party graphics development tool called Dr. Halo, but give no ordering information. (I discovered elsewhere that Dr. Halo, and an associated scanner, are available from Diamond Flower Electric Co. of Sacramento, California.)

The technical references are quite good. Liberal examples show you how to use each element discussed. The indexes, page headings, and tables of contents make the manuals easy to navigate. However, I did not like TI's use of spiral bindings for some of the manuals and the overstuffing of some of the three-ring binders. In my experience, spiral bindings make a manual impossible to maintain, and overstuff binders make it easy to tear the pages.

Single commands allow you to install and start each module. You will have no trouble here. TI has done a good job of documenting and simplifying both installation and initiation. Be careful to follow the instructions to the letter.

On- and off-site training is readily

available to all TI customers. Be sure to work through the user's guide before going to any of these workshops. A help desk is available for three months after your purchase of PC Plus. You can extend this period by paying an additional fee. I used this help desk and I must confess that I was a bit disappointed. TI does not provide a toll-free number during the initial three-month period. When I called, the answering machine kept me on hold for one and a half minutes. The call was then picked up by a data-entry person who asked me a host of questions, all but two of which had nothing to do with my problem. Then he gave me a "sequence number" and said a technician would return my call. This first call lasted 10 minutes and produced no results. The technician did call back that day. The question I had was a real curve ball. The technician handled it well and gave a correct answer. This person really knew what he was talking about.

Syntax. The syntax for PC Plus is simplicity itself. For the most part, unless you write directly in Scheme, you need not learn a special syntax. You must be conscious of only a few things, such as the use of the equal sign. On the IF side of a rule, "=" is a logical operator. On the THEN side of a rule, "=" is an "is-replaced-by" operator.

You must also pay attention to the type of parameter (or variable) used. For instance, some are Boolean (can be true or false), some are multivalued (can have several values at once), some are single-valued (can have only one of several values at once), and some are numeric values of limited range. How special functions operate depends on the type of parameter they are given. The differences are important, but can be subtle.

AND and OR logical relationships are interpreted through a hierarchy of

Table 1. Personal Consultant's modules and their minimum hardware requirements.

	PC Easy	PC Plus	PC Online	PC Images
Hardware	TI Business-Pro, IBM PC, PC-XT, PC-AT, PS/2, or true compatible	TI Business-Pro, IBM PC-AT, PS/2, or true compatible	TI Business-Pro, IBM PC-AT, PS/2, or true compatible	TI Business-Pro, IBM PC-AT, PS/2, or true compatible
Base Memory	512 Kbytes	640 Kbytes	640 Kbytes	640 Kbytes
Extended/Expanded Memory	—	—	512 Kbytes	512 Kbytes
Graphics Adapter/Monitor	CGA or EGA	CGA or EGA	CGA or EGA	EGA only
Recommended Configuration	640-Kbyte memory Hard disk EGA	1-Mbyte expanded memory EGA	1-Mbyte expanded memory EGA	1-Mbyte expanded memory EGA

operations or through standard parenthetical statements. If you have written in a common procedural language such as Basic, Fortran, or Pascal, you will have no trouble constructing AND/OR statements that reflect your meaning.

Implementing with PC Plus. Implementing the Statistical Strategist in TI's Personal Consultant Plus was quite an interesting experience that resulted in a 27-rule application. The original implementation was based on a non-rule, forward-chaining concept. PC Plus uses rules and a backward-chaining concept that does not easily lend itself to the original's directed binary inferencing. It also requires a strict definition of parameters and goals not originally required. However, the enforcement of this discipline resulted in some added power to the application.

User help facility. Without writing in Scheme, you cannot implement the generic help facility of the original application. It is also not a simple task to capture a generic user input such as "help," call out to an external procedure that will perform the appropriate operations, and return to the conversation where you left off.

However, TI designed PC Plus to offer specific help. In this regard, they have really done a fantastic job. The knowledge engineer can associate specific help information with each parameter and rule. The user easily accesses this information during a conversation by pressing the F1 and F2 keys and selecting the Why and How options. (Why tells why a given question is being asked; How tells how a given conclusion was reached.) In this way you can offer the user literally thousands of explanations.

Easy knowledge base entry. By providing a sophisticated environment, TI has made it easy for a knowledge engineer to enter, maintain, and improve knowledge bases. PC Plus prompts you for the IF and THEN parts of each rule. It also prompts you for the information required to initiate a new application. It keeps track of a number of details to ease the workload and permit you to concentrate on the application. For instance, when you make a change anywhere in the knowledge base, all implied structural changes are made automatically.

I could not bypass the PC Plus environment to simply enter domain knowledge directly. You must provide the interface between the domain expert and PC Plus.

Three quirks about text entry are worth mentioning here.

(1) The Home key toggles the cursor between the beginning and end of the line. It would be more natural for the Home and End keys to move the cursor to the appropriate line positions.

(2) The cursor moves from beginning to end very slowly.

(3) No command will let you automatically center a line of text.

These three quirks make text-screen development and maintenance a little tedious.

Interactive conversation that makes no initial assumptions. Personal Consultant generally presents a good interactive interface to the end user. It assumes that all goals must be proven, disproven, or found to be undeterminable. Thus, it can spend a lot of time asking questions about goals that will never be proven. The knowledge engineer can keep this from becoming a problem by placing the parameter that tends to be

the biggest differentiator at the head of the list of parameters in the goal rules.

For instance, I have a rule group that determines the purpose of the statistical calculation to be performed. Once that purpose is determined, the system ignores all questions concerning methodologies that do not meet that purpose. You can do this by having a trigger parameter that is set when a purpose parameter is determined. Then, no other questions about such parameters are asked. The remaining purposes are undetermined, leading directly to undetermined goals (methodologies, in this case).

Machine translation of the knowledge base. PC Plus is a tool for knowledge engineers, not for domain experts. It provides no environment for the domain expert to enter information for later translation. Since TI provides an interface to Scheme, you could write a domain expert interface to support knowledge capture at that level. You could then translate the domain expert's input into Scheme or the disk-storage version of ARL (TI's Abbreviated Rule Language). However, the manuals don't explain the disk-storage version of ARL. While they do explain Scheme in detail, it is a low-level language requiring a large programming effort for automatic code generation.

Logic trace facility. You will find that PC Plus provides a good toolkit for inference tracing. All details of the inferencing process are shown during a trace. Especially useful is the display of the entire effort to backtrack to a parameter value that can be directly determined. This is followed by the trace of that parameter's use during the search for secondary parameter values.

You can put the trace in a file, on the screen, or on the printer. You can have it start or stop at any user-input point in the conversation.

Using the trace facility, I had no trouble observing the inference process caused by my knowledge base. In fact, I learned a lot about backward chaining from watching it.

Restart capability. Again, I found this an easy task. PC Plus makes a command menu available at all user-input points in the conversation. At any time, the end user can ask for a restart, trace, explanation, review, or end to the session.

PC Plus permits you to change answers within the conversation after the conversation has taken place or even in the middle of the conversation. The original application does not permit this. Restarting a conversation with this kind of "what if" game playing gives you quite a powerful feature.

No time delay between questions and recommendations. Since Personal Consultant backward chains from the goals list, you can expect some variable delay as the conversation continues. PC Plus uses the facts it has gathered from the end user to determine other facts and to explore goals that may or may not be applicable. The more facts that are determined, the less it has to converse with the user. I found it disquieting that no "Inferencing" message appeared on the screen during times when PC Plus went off on its own. As a developer, you can employ meta rules to force the program to execute the most useful rules first. This can speed up the inferencing process.

Another reason behind the delays between questions is that PC Plus does a lot of software- and knowledge-base overlaying and garbage cleanup (if less than full extended memory is available). It has to go to the disk numerous times, even if main memory is fully populated. The initial loading of the Statistical Strategist and the starting of the conversation took 23 seconds.

PC Plus also takes an undue amount of time to come to even trivial conclusions. It wants all available information before displaying even conclusions that use only one piece of that information. It should display conclusions as it comes to them and let the user decide if it is worth going on with the conversation. Of course, the final listing of all conclusions makes a useful report. By using the Print and Show functions and appropriate rules, the developer could force the display of conclusions as the conversation progresses. However, this

violates TI's apparent philosophy that the shell should handle such details.

One last point about the conclusions report: PC Plus does not generate the report if it has made no conclusions and if the developer set the property that eliminates reports on goals not met. The developer can get around this with a rule that becomes true if all goals are false or not determined. However, you must remember to update that rule every time a goal is added.

PC Plus wants all available information before displaying even conclusions that use only one piece of that information.

A further source of delays arises from the function that rules can use to execute an external program. If the developer does not specify how many paragraphs of memory PC Plus should transfer to disk in order to make room for the program, PC Plus transfers the entire executing memory to disk and then back in again when execution of the shell is to resume. This takes far too long. In the case of a missing memory transfer value, PC Plus should figure out for itself the size of the external program. Only in the case where the external program makes dynamic use of memory should the user have to figure out how much memory to transfer. As an application changes, it can be difficult to keep track of all the changes that you must make to calls for external programs. The program could still make total memory transfers if it allowed the developer to specify a value of "All."

Debugging capability. As I mentioned earlier, PC Plus's trace, playback, and review capabilities make logic debugging a snap. The syntax debugging is not so friendly, however. For example: The program prompts you for the IF and THEN parts of each rule. Should you actually type in "if" or "then," PC Plus generates the cryptic error message "ERROR: Possible invalid use of \$AND." I spent quite some time trying to deduce what this error message meant, especially the reference to \$AND. Finally, I found \$AND in the appendix to one of the manuals. It did not appear in any index.

The information did not help. Intuition eventually said that if the system prompted for IF and THEN, you should not actually type in those words. To be fair, the user's and reference manuals' examples always have "IF" and "THEN" in a color other than that indicating user input.

More about Personal Consultant. I did not use the Images or Scheme modules to implement the Statistical Strategist. However, I did look at both.

PC Images. Currently in version 1.0, PC Images offers the powerful concept of using graphics instead of text for user input. For instance, instead of getting the user to type in the position of a machine's switch settings, the program presents a picture of the machine. The user moves a cursor from switch to switch and setting to setting. This is potentially much easier than trying to get the user to understand a text description of the machine and its switches.

You can also use PC Images to create an on-screen data entry form that the user fills in instead of answering a lot of questions. From running the example included with PC Images, I gather that you cannot allow the user to move a picture's cursor "backward." If the user makes an error, he or she must retrace the conversation from the beginning to change the appropriate answer. This amounts to a lot of work just to correct a typing error.

The text data-entry forms have a wraparound feature, but the demo did not allow access to all the elements on the form.

Another problem is that, if the user selects "Continue" after a consultation (in which case a "New Start" is supposed to occur), the initial data-entry screen is only partially displayed.

One last point: You will find it difficult to set very small gauges because of the resolution limitations of EGA graphics. So, you will need to watch out for some image size limitations.

All in all, PC Images is a good idea, but version 1.0 needs some work.

PC Scheme. PC Scheme, TI's Lisp implementation, is now in version 3.0. PC Plus is written in Scheme, so it's easy to integrate Scheme modules into an application. The ability to link developer-defined modules directly into a generic shell is an important improvement in the field. Previously, developers would start with a generic shell and then have to implement their applications again in a custom shell due to the limitations of the generic shell.

Summing up Personal Consultant Plus. PC Plus could not directly implement every detail of the Statistical Strategist without some programming in Scheme (to do the generic Help facility and domain expert knowledge input and quality control). However, the power of the application increased because of the parameter discipline enforced by the development environment. The original Statistical Strategist has no knowledge of "purpose,"

"population size," or the meaning of other parameters. It simply has data tokens that can be either true or false. Therefore, in a backward chaining, rule-oriented environment, many duplicate questions can be asked. For instance, a data token that involves a population size greater than 30 is not necessarily related to one that involves a size greater than 20. Personal Consultant wants to know the type of data token so that a population size greater than 30 is automatically greater than one of 20. The definition of data token types permitted PC Plus to identify methodologies that would not be identified by the original Statistical Strategist.

Personal Consultant takes considerable commitment for you to learn to apply it effectively. Once this learning process reaches its critical mass, however, Personal Consultant is a powerful expert systems development and delivery tool.

Reader Service 25

Recommendations

Both Clips and PC Plus proved themselves worth the effort needed to learn them. They have their limitations, but they are solid knowledge-engineering tools that permit you to start with a small learning effort and go on to develop major applications involving heavy customization. I would recommend either depending on the developer's specific needs.

As a closing note, I would like to thank the following people for their help: Ed Alexander and Jim Krug of the Air Force Armament Laboratory; Richard Eckhouse of MOCO, Inc.; and Marilyn Raeth of Humana, Inc. I also appreciate the able support of Steve Baudenistel of Computer Science Corp. and Debbie Adams and Isam Khouri of Texas Instruments.



**NANYANG
TECHNOLOGICAL INSTITUTE**
Singapore

TEACHING APPOINTMENTS

The Nanyang Technological Institute, fully supported by the Government of Singapore, is one of the two institutions in the Republic that provide university level education. There are, at present, four Schools in the Institute viz. the School of Accountancy, the School of Civil & Structural Engineering, the School of Electrical & Electronic Engineering and, the School of Mechanical & Production Engineering.

Due to our rapid expansion programme and our aim to achieve the staff/student ratio of 1:10 speedily, we are currently recruiting staff especially those who specialise in the following areas:

SCHOOL OF ACCOUNTANCY

1. Financial Accounting
2. Cost and Management Accounting
3. Auditing and Information Systems
4. Legal Studies and Taxation
5. Business Communication and English Proficiency
6. Management, Organisational Behaviour and Human Resource Management
7. Business Statistics and Quantitative Methods
8. Principles of Economics, Monetary Economics and Public Finance

SCHOOL OF ELECTRICAL & ELECTRONIC ENGINEERING

1. Microprocessors
2. Computer Interfacing
3. Real-Time Software
4. CAD/CAM Tools
5. Data Base/Data Communications
6. Artificial Intelligence and Expert Systems
7. Microelectronics - Thick and Thin Film Technology
 - Optoelectronics
 - VLSI
8. High Frequency Circuit Design
9. Communication Networks and Systems
10. Communication Software
11. Digital Signal Processing including Image and Speech
12. Radio Propagation and Systems
13. Power Electronics and Drives
14. Intelligent Robotics Systems
15. Power System and Energy Management

SCHOOL OF MECHANICAL & PRODUCTION ENGINEERING

1. Electro-Mechanical Control Systems
2. Engineering Production
3. Mechanical Design
4. Robotics and Microprocessing
5. Industrial Automation and Control

Qualifications

Candidates should have -

- a) at least a Master's degree in the relevant fields and, for those applying to teach accounting subjects, relevant professional qualifications in addition will be essential;
- b) sound professional/teaching experience in their areas of specialisation.

Gross annual emoluments range as follows:

Lecturer	: \$47,638 - \$58,685
Senior Lecturer	: \$53,161 - \$94,097
Associate Professor	: \$82,432 - \$113,539
Professor	: \$101,874 - \$134,536
(US\$1 = S\$2.04 approximately)	

The commencing salary will depend on the candidates' qualifications, experience and the level of appointment offered.

Leave and medical benefits will be provided. Depending on the type of contract offered, other benefits include: an end-of-contract gratuity or provident fund, a settling-in allowance, subsidized housing, education allowance, passage assistance, baggage allowance for the transportation of personal effects to Singapore and car loan.

The Institute encourages its staff members to undertake outside consulting work of a specialist nature. They are permitted to earn and retain such consultation fees up to a maximum of 60% of their gross annual emoluments in any one calendar year.

Candidates wishing to be considered should write to -

**THE REGISTRAR
NANYANG TECHNOLOGICAL INSTITUTE
NANYANG AVENUE SINGAPORE 2263**

giving their curriculum vitae and the names and addresses of three referees.

NEW PRODUCTS

Contact or send releases to Nancy Hays, Computer, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720, Compmail+ , n.hays

Accel revamps Tango-PCB and Tango-Route CAE/CAD software with Series II

Accel Technologies has revamped its Tango-PCB and Tango-Route CAE/CAD software packages. According to the company, a major change in the Series II versions shows up in the user interface, which has changed from a command-driven menu to a two-layer pop-up menu with dialog boxes, user-definable macros, and macros emulating the current command set.

The company has expanded the documentation offered with Tango-PCB and Tango-Route Series II, adding a reference section, command summary, and menu map to each. Help facilities have gone from on-line to an on-screen prompt line with on-line, context-sensitive help screens with an index.

Tango-PCB Series II permits the design of 32 x 32-inch boards. The total number of layers possible in the design is 19. A three-grid system (Snap, Visible, and Relative) permits user-definable increments from 1-1,000 mils. Track sizes are user-definable from 1-255 mils. Users can define pad sizes from 1-4,095 mils and hole diameters from 1-255 mils. Possible shapes include round, elliptical, rectangular, rounded rectangular, oval, mounting hole, and target.

Other Tango-PCB Series II features include the new ability to overlap components; support for more graphics protocols, printers, and plotters; the ability to create draft or final artwork for more design elements; and the abil-

ity to generate more reports.

Tango-PCB Series II costs \$595, including one year of updates, the quarterly newsletter, and technical support.

Tango-Route Series II shows similar revamping of its capabilities, the most obvious being the user-definable track and pad sizes, multiple track widths, a design rule check that covers six signal layers, and support for additional graphics protocols.

Tango-Route Series II costs \$495, including one year of updates, the quarterly newsletter, and technical support.

A combination package of both Series II, PCB and Route, costs \$995.

PCB: Reader Service 30
Route: Reader Service 31

Apple offers new Mac models—SE and 68030-based IIX

Apple Computer has announced the Macintosh IIX computer, based on Motorola's 16-MHz 68030 microprocessor and 68882 math coprocessor.

According to the company, the IIX is the first Macintosh with a 3½-inch floppy disk drive that can read and write to MS-DOS and Apple II formats.

The Macintosh IIX comes with 4 Mbytes of RAM, 256 Kbytes of ROM, color, graphics, sound, and NuBus. It uses the small computer systems interface, which allows daisy-chaining of up to seven peripheral devices.

The Macintosh IIX comes in two configurations. With an 80-Mbyte hard disk, mouse, System 6.0.2, HyperCard, and documentation, the Mac IIX costs \$9,369. With a 1.44-Mbyte FDHD (Floppy Disk High Density) floppy disk drive, the Mac IIX costs \$7,769.

Apple has also introduced a new configuration of the Macintosh SE. The new model features 2 Mbytes of RAM and an internal 40-Mbyte hard disk drive. Like other SE models, it has an 8-MHz Motorola 68000 chip and an



The Apple Macintosh IIX features the 1.44-Mbyte Floppy Disk High Density drive, which can read, write, and format disks in MS-DOS, OS/2, or Apple II ProDOS formats.

expandable SE-bus slot. The 2-Mbyte Mac SE costs \$5,069.

IIX: Reader Service 32
SE: Reader Service 33

IBM fleshes out ES/9370 family with new models

IBM has announced three new models of the Enterprise System/9370 family.

The 9373 Model 30 reportedly doubles the processor performance of the Model 20 and serves as an upgrade path, using the same cabinet as the Model 20.

The 9375 Model 50 reportedly doubles the Model 40's performance. It comes in a rack-mounted configuration.

Models 30 and 50 use IBM's CMOS chip technology. According to the company, this implements the System/370 architecture on a single processor card.

The 9377 Model 80 reportedly delivers about one and a half times the performance of the Model 60. It employs the same Thermal Conduction Module technology as the Model 90.

Prices range from \$37,000 for the Model 30 to \$70,000 for the Model 50 to \$142,000 for the Model 80.

Reader Service 34

Apple replaces its IIC with new IIC Plus

Apple Computer has replaced the Apple IIC with the new Apple IIC Plus. The new machine uses an 800-Kbyte internal disk drive for 3½-inch floppy disks. Users can choose 1- or 4-MHz operating speeds.

Other new features include an internal power supply, a security lock option, a slide volume control, and a revised pin configuration on the serial ports.

The Apple IIC Plus uses a 65C02 microprocessor with 128 Kbytes of RAM. Internal memory expands to more than 1 Mbyte. According to the company, the new model will run software for Apple IIe and IIC models. Programs in the 5¼-inch format will run on the Apple IIC Plus through a 5¼-inch disk drive connected to the computer.

The Apple IIC Plus costs \$675. The color system, including the CPU, color composite monitor, and monitor stand, costs \$1,099. The monochrome system, including the CPU, monochrome monitor, and monitor stand, costs \$859.

Apple has also announced an update to its system software for the Apple IIGS. The Apple IIGS System Software 4.0 includes a 16-bit operating system. The new operating system is reportedly compatible with ProDOS 16 and ProDOS 8.

The company has revised the Finder application and added the Advanced Disk Utility and the Installer utility.



The Apple IIC Plus replaces the IIC as the entry-level Apple II.

The Apple IIGS System Software Version 4.0 requires an Apple IIGS with 512 Kbytes of RAM, ROM version 01, and one 800-Kbyte floppy disk drive. Hard disk users also need ROM Rev. C on the Apple SCSI card.

The Apple IIGS System Software Version 4.0 package includes two disks containing GS/OS, the Finder, the Advanced Disk Utility, and the Installer, as well as two manuals. It costs \$39.

IIC Plus: Reader Service 35
4.0: Reader Service 36

HP ships 33-MHz 68030-based workstation

Hewlett-Packard offers the HP 9000 Model 370, a workstation based on the 33-MHz Motorola 68030 processor. According to the company, the new workstation achieves a performance of 8 MIPS.

Hewlett-Packard has also announced a new floating-point accelerator, the FPA, for the Series 300 workstations. The company reportedly achieved benchmarks for the FPA of 730 double-precision KFlOps on a Model 370. The Floating Point Accelerator costs \$3,500.

Model 370 workstations include a 32-bit system bus, the IEEE 488 peripheral interface, direct memory access, and Ethernet/IEEE 802.3 Thin-LAN or attachment unit interface for network access. The standard configuration includes 8 Mbytes of parity RAM, expandable to 32 Mbytes.

The company's standard operating

system, HP-UX 6.2, includes licenses for C, NS-ARPA, NFS, and the X-Window system.

The Model 370 comes in six basic configurations. The Model 370 SPU, an entry-level system, costs \$21,900. The Model 370 MH comes with a monochrome monitor for \$24,500. The Model 370 C+, which features six planes of 2D color graphics, costs \$25,000. The Model 370 CH, a 2D color-graphics workstation with 10 planes, costs \$31,900. The Model 370 SRX costs \$50,250 for 3D solid-rendering color graphics with eight planes expandable to 24. The Model 370 TurboSRX features 3D solid-rendering color graphics with 24 planes and a 16-bit Z buffer for \$72,250.

Model 370: Reader Service 37
FPA: Reader Service 38

AMT's DAP 610 employs 4,096 processors

Active Memory Technology has added the DAP 610 to its Distributed Array of Processors family of parallel computing systems. The new system employs a 64 × 64 matrix of 4,096 processors.

The DAP 610 reportedly delivers 40 billion Boolean operations per second, four times the rate of the DAP 510. According to the company, the system can multiply integers at a rate of 400 million per second in a 16-bit environment.

The new system includes a high-speed data channel, connectivity with DEC VAX computers and Sun Microsystems workstations and servers, software development tools, and software libraries.

The DAP 610 with 16 Mbytes of memory costs \$320,000.

Reader Service 39

Modular OS-9 handles many users, many tasks

Radstone Technology (formerly Plessey Microsystems) offers an operating system called OS-9, which the company calls a modular multiuser operating system for 68000 family VMEbus processors. OS-9 is available for professional development environments and for ROM-based industrial systems.

Professional OS-9 is a multiuser development environment for 68000 or 68020 disk-based systems. It includes a real-time kernel; disk, tape, pipe, and character file managers; utility set; screen editor; C compiler; documentation; and 90-day, end-user hotline support from Microware.

Industrial OS-9 targets ROM-based, embedded real-time applications. It includes a real-time kernel, character and pipe managers, and associated drivers for industrial applications.

Other high-level languages available for OS-9 include Pascal, Fortran, and Basic.

Professional OS-9 costs \$900 for 68020 systems and \$600 for 68000 systems. Industrial OS-9 costs \$300 for 68020 systems and \$200 for 68000 systems.

Professional: Reader Service 40
Industrial: Reader Service 41

Intel expands real-time products with 386 boards, Multibus II controller, computer

Intel has announced a variety of new real-time products, including a second-generation series of 80386-based CPU boards for the Multibus I architecture, a peripheral controller for Multibus II systems, and a computer using the company's new Multibus II system architecture (MSA).

The 32-bit iSBC 386/12 CPU boards are based on the 80386 microprocessor. According to the company, the boards offer new I/O capabilities, including two serial ports, one parallel port, two SBX connectors, a DMA controller, and an iLBX bus connector. A custom memory controller reportedly increases memory addressing flexibility. The addressing scheme allows the boards to address memory above the 16-Mbyte limitation of the Multibus I architecture. Dual-port arbitration control logic and memory aliasing provide multiprocessor support.

The iSBC 386/12 boards come in two speeds, 16 and 20 MHz; in four memory configurations, 1, 2, 4, and 8 Mbytes; and with or without an 80387 numeric coprocessor. Prices start at \$3,995 for a 16-MHz, 1-Mbyte board without coprocessor and range to

\$10,545 for a 20-MHz, 4-Mbyte board with coprocessor.

The iSBC 386/258 peripheral controller uses a 16-MHz 80386 processor and a cache of 1 or 4 Mbytes. It supports single-ended and differential small computer systems interface peripherals. It supports data rates up to 1.5 Mbytes per second asynchronously and up to 4 Mbytes per second synchronously. The controller has an SBX connector and a serial I/O port supporting data rates from 300 to 19,200 baud.

The iSBC 386/258 with 1 Mbyte of cache costs \$3,995. Shipments are scheduled for December.

The Multibus II system architecture is, according to the company, a hierarchy of hardware, firmware, and software interfaces and protocols. It relies on the concepts of scalable multiprocessing for high performance, the Multibus II transport protocol and message passing, multiprocessor system boot and initialization, and system and board diagnostics.

The 386-based System 520, which uses MSA, expands to accommodate four processors. It runs the iRMX II real-time operating system and features

a small-computer-systems-interface peripheral controller and a windowed-graphics, virtual terminal I/O server.

The System 520 comes in three configurations. The base original-equipment-manufacturer system features an eight-slot floor chassis, an iSBC 386/258 SCSI peripheral controller, a central services module, a 540W power supply, a floppy disk drive and controller, and documentation. The base plus I/O OEM configuration adds an iSBC 386/120 CPU board with 4 Mbytes of DRAM, an iSBX 279 graphics controller, a terminal controller, an Ethernet controller, a 380-Mbyte hard disk drive, and a 150-Mbyte tape drive. The iRMX II development system configuration adds iRMX II development system software and hardware and software support.

The System 520 is scheduled for availability in January 1989, with prices starting at \$11,130. It comes in floor-stand or table-top packaging.

iSBC 386/12: Reader Service 42
iSBC 386/258: Reader Service 43
System 520: Reader Service 44

AST Research adds to its Premium/286 PC and workstation lines

AST Research has added new models to its Premium/286 personal computer line and its Premium Workstation/286 line.

The AST Premium/286 Model 140V features AST-VGA Plus, a 16-bit graphics adapter; 10-MHz speed with zero-wait-state operation; 1 Mbyte of RAM; a 40-Mbyte hard disk drive; one

serial port; one parallel port; and seven expansion slots, including two AST Fastslots. The Model 140V costs \$4,095.

The company also offers a new memory expansion board for use in Premium/286 Fastslots. Advanced FastRAM provides zero-wait-state operation. It comes with 2 Mbytes of mem-

ory expandable to 8 Mbytes on one board. The board supports the Expanded Memory Specification 4.0. It costs \$1,745.

Two new models join the 10-MHz Premium Workstation/286 line. The 123X and 125X feature a 28-ms, 20-Mbyte hard disk drive. The 123X has a 3½-inch, 1.44-Mbyte floppy disk drive, and the 125X has a 5¼-inch, 1.2-Mbyte floppy disk drive.

Standard features include 512 Kbytes of RAM with EEMS/EMS 4.0 support; two full-height, 8-/16-bit expansion slots; two serial ports; a bidirectional parallel port; clock/calendar; 101-key keyboard; software cache; support for an 80287 math coprocessor; and a system board connector for an optional video module, floppy disk controller module, and fixed disk controller.

Models 123X and 125X cost \$2,145. Optional VGA, EGA, and monochrome modules cost \$300, \$300, and \$145, respectively.

140V: Reader Service 46
FastRAM: Reader Service 47
123X: Reader Service 48
125X: Reader Service 49

MacTCP integrates Mac in multivendor environments

Apple Computer offers MacTCP network software, based on the Transmission Control Protocol/Internet Protocol standard. According to the company, the software lets Macintosh PCs operate and share information with different computer systems.

MacTCP runs on Macintosh computers from the 512Ke to the 11x over Ethernet- and LocalTalk-compatible cable systems. Although coexistent with AppleTalk network protocols, the software has full access to AppleTalk services. According to the company, it can acquire network addresses dynam-

ically without user intervention and uses the Macintosh control panel to simplify address configuration procedures.

MacTCP includes the IP, TCP, UDP, ICMP, ARP, RARP, RIP, and BootP protocols. It is a site-licensed product for developers. An internal use license will cost \$2,500 and a commercial license will cost an additional \$2,500, with discounts for higher education institutions. MacTCP is scheduled for availability in the first quarter of 1989.

Reader Service 45

FOURTH CONFERENCE ON HYPERCUBE CONCURRENT COMPUTERS AND APPLICATIONS

MARCH 6-8, 1989

HYATT REGENCY MONTEREY'S CONVENTION CENTER MONTEREY, CALIFORNIA

Program Committee

Cil Weigand
General Chairman
Sandia National Laboratories
John Gustafson
Program Chairman
Sandia National Laboratories
Bill Hickey
Conference Administration

Organizing Committee

Don Austin
Department of Energy
Tony Chan
University of California
at Los Angeles
Terry Cole
Jet Propulsion Laboratory
Erik DeBenedictis
ATT/Bell Labs
Geoffrey Fox
California Institute of Technology
Bill Groppe
Yale University
Michael Heath
Oak Ridge National Laboratory
Paul Messina
California Institute of Technology
Ed Oliver
Air Force Weapons Lab
Quentin Stout
University of Michigan



HCCA4 Host Institution
Sandia National Laboratories

Sponsors

U.S. Department of Energy/Applied
Mathematical Sciences Program
Strategic Defense Initiative
Organization/Offices of Innova-
tive Science and Technology
Joint Tactical Fusion Program Office
U.S. Air Force Electronic Systems
Division
Air Force Office of Scientific Research
NASA

PURPOSE:

The aim of the Fourth Conference on Hypercube Concurrent Computers (HCCA4) is to exchange information on the fast-developing area of parallel computers with distributed memory. This includes, but is not limited to hypercubes, meshes, and massively parallel ensembles of single-bit processors.

SESSIONS:

Technical sessions will include invited talks, contributed papers and posters, and panel discussions. Sessions will address a wide range of distributed memory computer issues in areas of hardware, software (both systems and tools), applications, programming environment, and parallelization methods. Session titles include:

Architectures	Graphics	Algorithms
Operating Systems	Debugging	Numerical Methods
Image Processing	Scientific Applications	Concurrent Languages
Programming Environments	Artificial Intelligence	Performance Evaluation
Non-numerical Applications	Load Balance/Mapping	Parallelization Tools

CONFERENCE REGISTRATION:

Complete and return this form to: HCCA4, P.O. Box 428, Los Altos, CA 94023.

	before 2/15/89	after 2/15/89
Conference (including Banquet and Proceedings)	\$200	\$250
Conference-Student (Proceedings only)	\$100	\$150
Additional Banquet	(\$30 per person)	

Make checks payable to HCCA4. Pre-registration by mail is available until 2/15/89. After 2/15/89 registration will be accepted at the conference only. Requests for refunds must be made in writing no later than 2/15/89.

AMOUNT ENCLOSED \$_____

☐ Check enclosed ☐ VISA ☐ MasterCard ☐ American Express

Card No _____ Signature _____ Exp. Date _____
Name _____
Organization _____ Phone Number _____
Mailing Address _____
City _____ State _____ Zip _____

HOTEL RESERVATIONS:

A block of rooms has been reserved until February 1, 1989 at the Hyatt Regency Monterey (Conference Hotel) and the nearby Monterey Hotel Resort. Attendees can make their own reservations by calling:

HYATT REGENCY MONTEREY (408) 372-1234 Single \$118 Double \$143
MONTEREY HOTEL RESORT (408) 373-6141 Single or Double \$66

For additional information or exhibit information please contact:
Bill Hickey, Conference Administrator, (415) 969-6920.

Compaq expands Deskpro line with 386/20e

Compaq Computer has added the Deskpro 386/20e to its line of 80386-based computers. The 386/20e incorporates a 20-MHz Intel 80386 microprocessor and Compaq's Flex architecture, which provides the 20-MHz Intel 82385 cache memory controller, a 32-bit memory bus, and a separate peripheral expansion bus compatible with 8- and 16-bit add-on boards and peripherals.

The 386/20e comes in three models.

Model 1 includes 1 Mbyte of 32-bit RAM, a cache memory controller with 32 Kbytes of static RAM, one 5 $\frac{1}{4}$ -inch 1.2-Mbyte disk drive, five available expansion slots (four 8-/16-bit slots and one 32-bit slot), a socket for a 20-MHz Intel 80387 coprocessor or Weitek 3167 coprocessor, and a video graphics controller. It also includes a 101-key enhanced keyboard; parallel, asynchronous communications and mouse interfaces; real-time clock/calendar; security lock; power supply; and limited warranty. Model 1 costs \$5,199.

Model 40 has the same features as Model 1, but adds a 40-Mbyte fixed disk drive. Model 40 costs \$6,599.

Model 110 has the same features as the Model 1, but adds a 110-Mbyte fixed disk drive. Model 110 costs \$7,999.



Compaq's Deskpro 386/20e measures 5.9 x 14.8 x 15.8 inches, taking up less work space than other Deskpro models.

All models include Compaq's Expanded Memory Manager software for applications that support the Lotus/Intel/Microsoft Expanded Memory Specification; disk cache software; and power-on and keyboard password security software.

Reader Service 50

Checkpoint automates measurement

Software Productivity Research offers a knowledge-based automated measurement tool called Checkpoint. The expert knowledge incorporated in the project management tool comes from company chairman Capers Jones.

The assessment feature compares a specific project to industry standards in terms of cost, quality, effort, and productivity, according to the company. Users can then conduct "what if?" experiments to compare alternative courses.

The estimation feature offers cost estimates, reputedly to an accuracy of five percent. Factors include staff size, quality requirements, development activities, environment variables, risks, and value analyses.

The measurement feature monitors the progress of programming activities and the factors that affect them.

The estimation feature uses update information provided by the measurement feature to adjust estimates.

Checkpoint sells for \$20,000. The price includes documentation and an on-site class introducing the product.

Reader Service 52

IBM adds new PS/2 model based on Intel's 80286

IBM has added an entry-level model to the Personal System/2 family with the Model 30 286. The new model comes with a 10-MHz Intel 80286 microprocessor, 512 Kbytes of RAM expandable to 4 Mbytes, 128 Kbytes of ROM, a 1.44-Mbyte 3 $\frac{1}{4}$ -inch drive, Video Graphics Array, three available full-size slots, memory expansion up to 16 Mbytes, and a 16-bit bus architecture. A 10-MHz 80287 math coprocessor is optional.

The system board integrates serial, parallel, keyboard, pointing device, floppy disk drive, and VGA graphics capabilities, plus a time and date clock with a lithium battery.

Two configurations are available: Model 30 286-E21, featuring a 20-Mbyte fixed disk drive with integrated controller, and Model 30 286-E01, without a fixed disk drive. The Model 30 286-E21 costs \$2,595 and the Model 30 286-E01 costs \$1,995.

The Model E21 weighs 19 pounds and



IBM's PS/2 Model 30 286 comes with VGA graphics and reportedly has twice the performance and disk drive capacity of the original Model 30.

the Model E01 weighs 17.2 pounds. Both measure 15.6 x 16 x 4 inches.

Reader Service 51

Coprocessor targets Mac II

Mercury Computer Systems' MC3200NU floating-point coprocessor targets the Apple Macintosh II computer. The single-slot card reputedly achieves a performance rate of 20 Mflops and 10 MIPS.

According to the company, the Macintosh Programmers' Workshop provides the development environment. The card is programmable in C, Fortran, assembly, and microcode. The card features a collection of microcoded algorithm routines.

The MC3200NU is based on Weitek's XL-8032 RISC processor chip set, supported by up to 8 Mbytes of two-way interleaved memory on a 40-Mbyte per second internal bus.

The MC3200NU can be both master and slave on the Macintosh II.

A 2-Mbyte system costs \$10,000 for the hardware only. The Fortran and C development environment costs \$8,500.

Card: Reader Service 53
Software: Reader Service 54

9th SYMPOSIUM on COMPUTER ARITHMETIC

ARITH 9

SEPTEMBER 6-8, 1989
Santa Monica,
California



Sponsored by the Technical Committees on Computer Architecture (TCCA)
and VLSI (TCVLSI), IEEE Computer Society

in cooperation with IFIP WG 2.5 and UCLA Computer Science Department

CALL FOR PAPERS

Authors are invited to submit papers describing new theoretical and applied results on all aspects of computer arithmetic, including but not restricted to the following topics:

- Mathematical Foundations of Computer Arithmetic
- Arithmetic Algorithms: Analysis, Implementation, and Design Methodologies/Tools
- Arithmetic Processor Architectures and Implementations
- Design of Testable and Dependable Arithmetic Systems
- Arithmetic Aspects in Application Areas such as Signal and Image Processing, Graphics, and Robotics
- Numerical Error Control and Error Analysis
- High Level Language Support of Arithmetic Systems

Four (4) copies of the complete paper (in English, not exceeding 20 double-spaced typed pages) should be submitted to Prof. Miloš D. Ercegovac, Program Co-Chair, no later than February 1, 1989. Since refereeing will be anonymous, author(s) identification should *only* appear on a separate sheet. Authors will be notified of acceptance by April 1, 1989, and final camera ready papers (up to 8 pages) will be due May 1, 1989. Proceedings will be available at the symposium.

General Chair

Prof. Algirdas Avizienis
Computer Science Department
4731 G Boelter Hall
UCLA
Los Angeles, CA 90024
Phone: 213/825-3028
e-mail: aviz@cs.ucla.edu

Program Co-Chair

Prof. Miloš D. Ercegovac
Computer Science Department
3732C Boelter Hall
UCLA
Los Angeles, CA 90024
Phone: 213/825-5414
e-mail: milos@cs.ucla.edu

Program Co-Chair

Dr. Earl Swartzlander
Defense Systems Group
Bldg. R2/1094
TRW
Redondo Beach, CA 90278
Phone: 213/812-0791

Conference Location: Loews Santa Monica Beach Hotel

IC Announcements

Company, Model, Function

Comments

R.S. No.

Advanced Hardware Architectures AHA4510 FEC codec	A programmable Reed Solomon forward-error-correction codec. Correction time independent of the error pattern and number of errors. Requires no external buffering. Has encoder and decoder modes. Comes in a 68-pin, J-lead PLCC. Cost (1,000s): less than \$100.	120
Advanced Micro Devices 67C450X FIFOs	Densities of 1/2-2K in speeds from 25-80 ns. Have multiplexed array internal architectures, can accept and output data asynchronously and simultaneously, and operate at speeds up to 28.5 MHz. Also have retransmit capabilities. Come in 28-pin plastic and ceramic DIPs and 32-pin PLCCs. Cost (100s): \$11-\$63.	121
Cylink CY512 Key management processor	A key management processor for cryptographic key management applications. Performs the special math functions for public key calculations. Takes less than 100 ms for exponentiation of 512-bit numbers. Comes in a 600 mil, 24-pin plastic DIP. Cost: \$80.	122
Gemini Technology VC-002 VGA controller	A video graphics array controller chip capable of displaying 256 colors simultaneously at a resolution of 640 x 480 and 16 colors at 800 x 600 resolution. Supports three memory configurations: one or two banks of eight 64K x 4 RAMs or one bank of eight 256K x 4 RAMs. Cost: \$45.	123
Integrated Device Technology IDT7M6028, IDT7M6029 SRAM modules	Synchronous static RAM modules arranged 64K x 8 (IDT7M6028) and 64K x 9 (IDT7M6029), featuring 25-ns access times, internal pipeline registers, and separate data inputs and outputs. Come in 42-pin ceramic DIPs. Cost (100s): \$316 (IDT7M6028) and \$354 (IDT7M6029).	124
Linear Technology LTC1050 Op amp	A chopper-stabilized op amp that includes on-chip sample and hold capacitors. Max offset voltage of 5 μ V, max offset voltage drift of 0.05 μ V/ $^{\circ}$ C, a DC to 10 Hz input noise voltage of 1.8 μ V (p-p), and a typical voltage gain of 160 dB. Comes in an 8-pin plastic, small surface mount, large surface mount, or ceramic DIP. Cost (100s): starts at \$2.25.	125
NCR 53C80-40 SCSI controller	A low-power version of the 5380 SCSI controller that doubles the data transfer rate, operating at 3 Mbytes/s for most SCSI applications. Features worst-case power consumption from 0.761-0.079W. Supports the ANSI X3.131-1986 standard. Packaging options include 40-pin DIP and 44-pin PLCC. Cost (1,000s): \$6.07.	126
Motorola MC68HC68T1 Clock	A real-time clock/calendar with a 32-word x 8-bit static RAM and a synchronous, serial, three-wire interface. Features buffered clock output for driving CPU clock, timer, colon, or LCD backplane. Comes in a 16-pin plastic DIP or 16-lead SOIC. Cost (\$00-999): \$3.41 (DIP) and \$4.44 (SOIC).	127
Motorola MCM6293/4/5 SRAMs	Static RAMs organized 16K x 4, with separate I/O and different combinations of on-chip I/O registers or transparent latches on the outputs. Chip or output enabled. All three come in 28-lead 300-mil plastic DIPs. MCM6293 also comes in a 28-lead plastic SOJ. Cost (100s): \$30.46-\$52.10.	128
Motorola MCM62350/1, MCM4180 SRAMs	A family of cache address tag RAM comparators and status bit RAMs, organized 4K x 4. MCM4180 includes an exclusive NOR comparator. MCM62350/1 also include an AND-OR-Invert mode. MCM62351 has an open drain Match output. MCM62350 has a programmable valid Match level. Cost (100s): \$20.85-\$31.20.	129
S-MOS Systems SRM 21256, SRM 22256 SRAMs	SRM 21256 is a 64K x 4 SRAM that comes in a 24-pin skinny DIP. SRM 22256 is a 32K x 8 SRAM that comes in 28-pin DIPs or SOPs. Cost (100s): for SRM 21256, \$80 (35 ns), \$65 (45 ns), \$55 (55 ns); for SRM 22256, \$25 (55 ns), \$20 (70 ns).	130
Texas Instruments TMS370C756 Microcontroller	An 8-bit microcontroller with 512 bytes of data EEPROM, 16 Kbytes of EPROM, and an A/D converter on-chip. The A/D module is an 8-bit converter with internal sample and hold circuitry. Comes as part of a sample kit (TMS370SK56), in a ceramic leaded chip carrier. Cost: \$100.	131
Texas Instruments T20C14 Microcontroller	A 16-bit digital signal processing microcontroller operating at 25.6 MHz. Comes with 256 Kwords of on-chip RAM and 4 Kwords of on-chip ROM. Can address 4 Kwords of off-chip memory. Uses a Harvard architecture. Has a 32-bit ALU and 32-bit registers. Production in mid-1989. Cost: around \$10 in volume.	132



IEEE TRANSACTIONS ON

KNOWLEDGE AND DATA ENGINEERING

MARCH 1989

VOLUME 1

NUMBER 1

PREMIERE ISSUE

A PUBLICATION OF THE IEEE COMPUTER SOCIETY



PREMIERE ISSUE
MARCH 1989
RESERVE YOURS TODAY!

AREAS ADDRESSED

- Acquiring and managing knowledge and data in developing and using information systems;
- Strategies for efficiently capturing and storing new knowledge and data.
- System modeling, design, access, security and integrity control mechanisms;
- Structures within centralized and distributed information systems providing increased intelligence and ease of use through multiple interface devices; and
- Methods to prolong the useful life of knowledge and data and its graceful degradation.

AREAS RECEIVING REGULAR, IN-DEPTH COVERAGE

- AI techniques and expert systems
- Knowledge and data engineering tools and techniques
- System architectures
- Algorithms and applications
- System integration and modeling
- Query, design, and implementation languages
- Integrity, security, and fault tolerance

To order your charter subscription to IEEE Transactions on Knowledge and Data Engineering, use the card at right or call (714)821-8380 and charge it to your MasterCard, Visa, or American Express Card.

- hard-disk drives for the Compaq 386/S. Come in formatted capacities ranging 144
200 Mbytes. Feature an average access time of 25 ms and a data transfer rate of 8
Cost: \$1,195 (40 Mbytes), \$2,289 (80 Mbytes), \$2,599 (100 Mbytes), \$3,756 (140
\$4,995 (200 Mbytes).
- backup system for laptop computers. Comes with a capacity of 40 Mbytes of for- 145
storage. Features a data transfer rate of 250 Kbits/s and adheres to the QIC 40
Uses the DC2000 tape cartridge. Plugs directly into the computer port. Cost:
- nal modem for the IBM PS/2 models 50, 60, and 80. Implements MNP Level 5 146
C error-correction protocols. Compatible with the AT command set. Includes a
ation file for automatic setup with PS/2 computers. Operates at 2,400, 1,200, or
Cost: \$649 with Crosstalk XVI software, \$549 without.
- er-optic modem with full-duplex data transport over optical fiber at 1.544 147
Available in an EMI/RFI Suppressed version. Features a composite error rate of
one error per 1 x 109 bits and a loss budget of 19 dBm with 100/140 μ m fiber.
470; \$2,470 for RFI/EMI.
- in board for IBM PCs and compatibles that combines a Video Graphics Array 148
h a frame grabber. Permits preview of images prior to capture without a second
Comes with 256 Kbytes of memory. Cost: \$699.



IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING



AIM

CALL FOR PAPERS

The new quarterly TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING aims to provide an international and interdisciplinary forum to publish results on the research, design, and development of knowledge and data engineering methodologies, strategies and systems. Targeted to serve researchers, developers, managers, strategic planners, users, and others interested in state-of-the-art activities in the knowledge and data engineering area. March 1989 will be the first issue.

ARTICLE GUIDELINES

ARTICLE SELECTION PROCEDURES

This new periodical is at the Transactions level. The selection of articles for publication will follow the guidelines used by other IEEE Computer Society Transactions, such as the IEEE Transactions on Software Engineering. A minimum of three reviews will be required for a decision to be made on each submitted or solicited paper.

TYPE OF ARTICLES PUBLISHED

This Transactions will publish original results in research and development in areas relevant to knowledge and data engineering. Papers that can be submitted for consideration include those that have not previously been published in another journal, or are not currently being published, as well as those that have been published in Conference Proceedings, Digests, and Records and that have undergone substantial revision. Invited papers from leading authorities in the knowledge and data engineering area will also be published. Three types of papers will be published:

- (1) Regular technical articles (25-35 double spaced pages, including figures, tables, and references):
 - (a) papers with extensive original results and
 - (b) in-depth surveys, which contribute to the understanding and advances in the knowledge and data engineering area;
- (2) Concise short articles (maximum 12 double spaced pages): papers with results that are important and original and are

presented in a concise form;

- (3) Correspondence articles (maximum 3 double spaced pages): comments on previously published articles, short extensions to current results, critiques on previous results, responses from authors, and corrections to previously published articles. An effort will be made to shorten the turnaround time for concise papers and correspondence articles.

GUIDELINES FOR SUBMITTING PAPERS AND PROPOSALS ON SPECIAL ISSUES

- (1) For invited papers and proposals for special issues, send 6 copies to:

C. V. Ramamoorthy, Editor-in-Chief
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720
ram@em.berkeley.edu

- (2) For all other submissions, send 6 copies of manuscript, complete with illustrations, abstract, and index terms, to:

Benjamin W. Wah, Associate Editor-in-Chief
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1101 West Springfield Avenue
Urbana, IL 61801
(217) 333-3516, (217) 244-7175
wah@aquinas@uuc.cso.uiuc.edu

Motorola
MCM6293/4/5
SRAMs

Static RAMs organized 16K $\times$ 4, with separate I/O and diff. I/O registers or transparent latches on the outputs. Chip or in 28-lead 300-mil plastic DIPs. MCM6293 also comes in a 2 (100s): \$30.46-\$52.10.

Motorola
MCM62350/1, MCM4180
SRAMs

A family of cache address tag RAM comparators and status. MCM4180 includes an exclusive NOR comparator. MCM62 OR-Invert mode. MCM62351 has an open drain Match out programmable valid Match level. Cost (100s): \$20.85-\$31.20.

S-MOS Systems
SRM 21256, SRM 22256
SRAMs

SRM 21256 is a 64K $\times$ 4 SRAM that comes in a 24-pin skinny SRAM that comes in 28-pin DIPs or SOPs. Cost (100s): for (45 ns), \$55 (\$5 ns); for SRM 22256, \$25 (\$5 ns), \$20 (70 ns).

Texas Instruments
TMS370C756
Microcontroller

An 8-bit microcontroller with 512 bytes of data EEPROM, A/D converter on-chip. The A/D module is an 8-bit converter hold circuitry. Comes as part of a sample kit (TMS370SK56). Cost: \$100.

Texas Instruments
T20C14
Microcontroller

A 16-bit digital signal processing microcontroller operating Kwords of on-chip RAM and 4 Kwords of on-chip ROM. C chip memory. Uses a Harvard architecture. Has a 32-bit ALU in mid-1989. Cost: around \$10 in volume.

IEEE copyright transfer form and general guidelines for submissions can be found in the January 1988 issue of IEEE TRANSACTIONS ON SOFTWARE ENGINEERING.

FURTHER INFORMATION

Any questions regarding the journal can be directed to either listed Editor.

Microsystem Announcements

Company, Model, Function	Comments	R.S. No.
Calera Recognition Systems TrueScan Scanner recognition card	A scanner recognition card for IBM PC-ATs and compatibles. Captures text and graphics in one pass. Automatically decolumnizes pages without intervention. Recognizes and formats paragraphs, tables, lists, centering, justification, indenting, bold, italic, and underline. Cost: starts at \$2,495.	135
Data Race MasterModem family Modems	Support Group III facsimile exchanges; IBM terminal emulation for 5250, 3270, 3770, and 3780 mainframe sessions; interaction at 9,600 bps with minicomputers; batch file transfer at up to 19,200 bps; and connection at 300, 1,200, or 2,400 bps to other modems or information services. Cost: \$995 for modem card, \$1,345 for external modem.	136
DEST WorkLess Station II Scanner	A multifunctional scanner with 300 dpi resolution and 8-bit gray-scale input of 256 levels of image data. Scans letter, legal, and A4 sizes of paper with weights ranging from 16-30 lbs. Has serial or SCSI connection. Uses OCR techniques of matrix matching, intelligent character recognition, and context processing. Cost: \$9,795.	137
DY-4 Systems DVME-138 Single-board computer	A 25-MHz, 68030-based single-board computer with the Advanced VMEbus Interface Chip Set for the VMEbus. Includes 4 Mbytes of dual-ported memory, two serial ports, sockets for standard 32-pin EPROMs, and three 16-bit counters. Cost: \$13,195.	138
Fastcomm Communications FDX 9624 Modem	A full-duplex, 9,600 bps modem compatible with V.22 bis, Bell 212A, and Bell 103 modems. Based on CCITT Recommendation V.32. Operates over PSTN or leased lines. Comes with a Hayes-compatible autodialer and a full set of Hayes commands. Cost: \$899.	139
General Micro Systems GMSV17 CPU board	A VMEbus CPU board based on the 33-MHz 68030 and a 68882 coprocessor. Has up to 1 Mbyte of dual-ported, zero-wait-state SRAM and 512 Kbytes of EPROM. Originates and services interrupts on the VMEbus. Features a 68155 Bus Manager and a Bus Master Boot. Cost (100s): starts at \$2,096.	140
Ideal Copier Division FS3012 Scanner	A 36-inch wide, 300 dpi scanner. Includes software to output seven raster file formats to floppy disk, hard disk, or tape. Splits large-format drawings into multiple-page facsimiles for transmission to other facsimile machines or by electronic mail. Cost: starts at \$12,500.	141
Maynard Electronics MaynStream 2200HS Tape backup system	A tape backup system for networking environments. Features a data transfer rate of 250 Kbytes/s, multitasking software, read-after-write error checking and automatic rewrite, error-correction code, and an effective head-to-tape speed of 150 inches/s. Cost: \$6,995 (PCs) and \$7,095 (PS/2s).	142
Mountain Computer Series 2100 Tape backup system	An external tape backup system with average data transfer rates of more than 13 Mbytes/minute using an SCSI interface. Compatible with AT-bus computers. Includes FileSafe software. Has Reed-Solomon error-correction code. Cost: \$6,495.	143
Procom Technology Propaq/S series Hard disk drives	Internal hard-disk drives for the Compaq 386/S. Come in formatted capacities ranging from 40-200 Mbytes. Feature an average access time of 25 ms and a data transfer rate of 8 Mbits/s. Cost: \$1,195 (40 Mbytes), \$2,289 (80 Mbytes), \$2,599 (100 Mbytes), \$3,756 (140 Mbytes), \$4,995 (200 Mbytes).	144
Procom Technology PLT40 Tape backup system	A tape backup system for laptop computers. Comes with a capacity of 40 Mbytes of formatted storage. Features a data transfer rate of 250 Kbits/s and adheres to the QIC 40 standard. Uses the DC2000 tape cartridge. Plugs directly into the computer port. Cost: \$750.	145
Ven-Tel 24/2E Modem	An internal modem for the IBM PS/2 models 50, 60, and 80. Implements MNP Level 5 and X.25 error-correction protocols. Compatible with the AT command set. Includes a configuration file for automatic setup with PS/2 computers. Operates at 2,400, 1,200, or 300 bps. Cost: \$649 with Crosstalk XVI software, \$549 without.	146
Versitron FOM9220 Modem	A T1 fiber-optic modem with full-duplex data transport over optical fiber at 1.544 Mbits/s. Available in an EMI/RFI Suppressed version. Features a composite error rate of less than one error per 1 x 10 ⁹ bits and a loss budget of 19 dBm with 100/140 µm fiber. Cost: \$1,470; \$2,470 for RFI/EMI.	147
Willow Peripherals Publishers' VGA VGA/frame grabber board	An add-in board for IBM PCs and compatibles that combines a Video Graphics Array card with a frame grabber. Permits preview of images prior to capture without a second monitor. Comes with 256 Kbytes of memory. Cost: \$699.	148

CALL FOR PAPERS

IEEE INTERNATIONAL CONFERENCE ON COMPUTER DESIGN: VLSI IN COMPUTERS & PROCESSORS

Sponsored by: **IEEE Circuits and Systems Society**
The IEEE Computer Society

In Cooperation with: **IEEE Electron Devices Society**

ICCD '89



Hyatt Regency Cambridge, Cambridge, Massachusetts
October 2-4, 1989



The International Conference on Computer Design: VLSI in Computers and Processors covers all aspects of the design and implementation of VLSI computer and processor systems. The multi-disciplinary nature of the conference is intended to emphasize the interactions among architecture, computer-aided design, design & test and VLSI technology.

Original papers are especially solicited in the following areas as they relate to computer and processor design:

Architecture and Algorithms: Computer Architecture: Concurrent Computers, Digital, Signal and Image Processors, Data Base Machines, Graphics Processors. Architectural Support for Operating Systems and Languages, Computer Design: Cache and Memory System Design, Processor Design, Computer Arithmetic, Computer Networks, Algorithms: Design and Analysis of Algorithms, Parallel Algorithms, Numerical Methods.

Computer-Aided Design: High-Level Synthesis, Silicon Compilation, Automatic Layout: Placement and Routing, Layout Verification, Timing Analysis, Logic and Circuit Simulation, Multiprocessor and Parallel Processor, Implementation of CAD Algorithms, Dedicated CAD Hardware, Integrated CAD Systems.

Design & Test: Mixed Analog/Digital Design, Design For Testability, Design For Self Test, Testing and Testability Analysis, Fault Modelling, Reliable Computing, Design For Reliability.

VLSI & Technology: State of the Art System Integration, Packaging, Wafer-Scale Integration, Environmental Factors, Optical Interconnects, VLSI: CMOS, Bipolar, GaAs, Low Temperature, Mixed Technologies, Storage: Optical, Magnetic, Semiconductor.

Papers are also solicited which describe innovative features of new products, and which describe the overall integration of Technology, Architecture, CAD, and Design and Test into the Computer Design Process.

Instructions to Authors

Prospective authors are invited to submit a 1000-2000 word summary. To be considered for review, summaries must include the following information: (a) clear description of the contribution and why it is important, (b) for original research submissions state what is novel about the contribution, (c) for review and tutorial submissions state the contribution to the multi-disciplinary mission of the conference.

Submit six (6) copies of summaries, along with the author names, addresses, office and home telephone numbers, by no later than

February 1, 1989 to Technical Program Chair: Sumit Das Gupta
IBM Corporation
Bldg. 306, ZIP 3A1
Hopewell Junction, NY 12533
Telephone: (914) 894-0540

- Proposals for specially organized sessions are solicited. Please forward your proposals to the Technical Program Chairman for review by no later than **January 2, 1989**.
- Summaries to Technical Program Chair: **February 1, 1989**
- Notification of acceptance: **March 31, 1989**.
- Final camera ready paper due **June 15, 1989**.
- Awards will be presented to the best conference papers.

PLEASE SEND ME MORE INFORMATION ABOUT ICCD '89.

NAME _____
COMPANY _____
ADDRESS _____
CITY/STATE/ZIP _____
COUNTRY _____

SEND TO: ICCD '89
1730 Massachusetts Avenue, N.W.
Washington, D.C. 20036-1903



THE IEEE COMPUTER SOCIETY



**THE INSTITUTE OF ELECTRICAL
AND ELECTRONICS ENGINEERS, INC.**

CONFERENCES

Editor: Edmund L. Gallizzi, Computer Science Dept., Eckerd College, St. Petersburg, FL 33733, (813) 864-8272; Compmail, e.gallizzi

Mead covers broad scope in neural networks talk at INNS annual meeting

Mark Kon, Boston University

Carver Mead of the California Institute of Technology presented a wide-ranging discussion of neural networks when he delivered a plenary talk at the International Neural Network Society's first annual meeting. More than 1,600 persons attended the Boston conference September 6-10.

Fourteen organizations from various fields (including the IEEE Computer Society and three other IEEE societies) cooperated with the INNS in conducting the conference. Bernard Widrow of Stanford University was the general chair.

In his talk, Mead said that complex neural systems maintain low levels of ambient activation by squelching input at low levels unless it is novel. With this notion in mind, Mead has constructed his artificial retina, which he described at the conference.

Though this silicon retina has a low pixel resolution, it is able to extract important information from its visual field by only reacting to novel stimuli. That is, it detects change, whatever that may be. He demonstrated the reactions of the retina when acclimated to uniform light and other types of visual stimuli.

Plenary presentations. Biological neural systems were the focus of a number of plenary presentations. Stephen Grossberg discussed a wide range of architectures he and colleagues at Boston University's Center for Adaptive Systems have studied and developed over the past several years.

One focus of Grossberg's talk was the preprocessing of the visual field. He described work on a pair of parallel but linked processing paths, the boundary contour and feature contour systems. The first generates boundaries of objects, and the second colors and fills in the contours.

A novel aspect of Grossberg's brain modeling is the surprisingly simple mechanisms he has found to implement the architectures he has studied. For example, the boundary and feature contour systems are hypothesized to implement a topographic map of the visual field in the brain, in which rapid diffusion of neural signals corresponding to flow of color within boundaries occurs, while boundaries themselves correspond to barriers where this diffusion is stopped. Grossberg's type of modeling has attracted a great deal of attention lately among scientists and engineers alike attempting to implement neural nets.

Interesting technological announcements included Widrow's description of a simulated broom-balancing device whose abilities are based on learning. The device sets as its goal to learn to balance a broom and, using network learning techniques, it develops the proper responses.

A physical device of this sort was demonstrated in the exhibit of Hecht-Nielsen Neurocomputer Corporation. This system initially learned balancing in a Macintosh simulation. Once its techniques were roughly defined, the network learned the real-world version of the task in about 12 hours; the machine's vision is provided by a camera. The network is implemented on HNC's Anza PC-compatible hardware.

Other research Widrow described included a mechanism designed to connect endings in nerve bundles severed when a person's hand or limb is detached. An electronic multiplexer is placed between the two endings of the severed nerve, and all possible connections among pairs of nerves are tested using a learning algorithm, with the gradual reestablishment of the proper connections.

A large number of vendors had exhibits demonstrating capabilities of

neural network simulation packages. These involve hardware and/or software personal computer packages capable of simulating networks with thousands of neurons, with user-specified connection weights. Learning algorithms are also included in these packages.

Among the many available simulation packages, Neural Ware demonstrated an application in which a simulated network looking at the behavior of stock market prices learns to make optimal predictions. Another product, MacBrain, by Neuronics, is a versatile simulator incorporating a number of currently popular network models. AI Ware demonstrated a control system in which the water level in a tank with variable cross-section, irregular output, and noisy detectors is varied randomly; the network makes the best use of information collected by sensors in order to maintain a constant water level.

President's address. Grossberg, the first president of the society, delivered a talk about the organization and the field of neural networks in general during the INNS meeting.

As of its first anniversary, Grossberg said, the INNS has become the fastest growing professional society in the world, expanding at a rate of about 200 members a month. It presently has about 3,500 members in 49 US states and 38 countries. Membership cuts across a number of fields, including computer and information sciences, engineering sciences, life and biological sciences, mathematics, physics, and business, he said.

The neural networks field also continued its phenomenal growth over the past year, according to Grossberg. The level of work in the area has become more sophisticated, and the amount and quality of supporting materials

(privately developed software and hardware, for example) has increased substantially.

In general, Grossberg predicted, neural networks will initially become important in decision making that is not amenable to good algorithmic analyses (otherwise it can already be handled by AI systems) in which very complicated multiple correlations exist among input and output variables. These can be

learned by networks, but would be difficult to pick out for analysts developing, for example, expert systems.

Grossberg cited such examples as weather prediction, evaluation of customer credit risks, and medical diagnosis and evaluation. The difference between the approaches is that neural networks can themselves determine the relevant variables, whereas traditional expert systems require a teacher's wis-

dom to guide them in perpetuity.

Widrow has been elected to succeed Grossberg as president of the INNS.

The society's second annual meeting is slated for September 5-9, 1989, in Washington, DC. Information on the meeting or on INNS membership can be obtained by contacting the INNS office, c/o J.R. Schuman Associates, 316 Washington St., PO Box 125, Wellesley, MA 02181, phone (617) 237-7931.

Symposium keynoter to focus on parallel computation model

Sushil Jajodia, George Mason University

A model for parallel computation that covers most of the cases of interest for programming and database systems, plus implementation of the unified model of parallel programming, will be described in the keynote address at the International Symposium on Databases in Parallel and Distributed Systems.

Slated December 5-7 in Austin, Texas, the symposium is cosponsored by the IEEE Computer Society Technical Committee on Data Engineering and the ACM Special Interest Group on Computer Architecture. Joe Urban of the University of Miami is the general chair, and Won Kim of MCC, Avi Silbershatz of the University of Texas at

Austin, and Sushil Jajodia of George Mason University are the program co-chairs.

J.C. Browne of the University of Texas at Austin and director of the University Research Initiative in Formulation and Programming of Parallel Computations will deliver the keynote talk entitled "A Unified Approach to Parallel Computations."

The model Browne will describe is intended to overcome the apparent diversity of conceptual elements for composing parallel computation structures, which is the stumbling block to attaining the benefits of parallelism.

The DPDS symposium has been

organized to focus attention on databases for parallel machines and heterogeneous distributed databases. Research in homogeneous distributed databases needs to be extended to these two areas.

A few commercial distributed database systems have recently been announced. However, most of the research has focused on homogeneous systems, a network of database systems supporting the same data model and language. Heterogeneous distributed databases remains a major research area.

Based on logic, functional, and object-oriented paradigms, a number of concurrent programming languages are being designed to better exploit the capabilities of parallel machines.

Researchers are beginning to understand the impact of the new programming paradigms on the data model and database system architecture, but considerable research remains to be done.

The symposium program will feature a single track to ensure high quality in technical content and a high degree of technical interaction during the paper sessions. Eighteen papers have been selected for presentation from among 64 submitted papers and three invited ones.

Two half-day tutorials will open the symposium December 5. They're entitled "Parallel Computing and Database Management" (given by Bruce Hillyer of AT&T Bell Laboratories) and "Integrating Heterogeneous Distributed Databases: Requirements, Concepts, and Solutions" (given by Ahmed Elmagarmid of Purdue University and Amit Sheth of Unisys).

For additional symposium information, contact Sushil Jajodia, ISSE Department, George Mason University, Fairfax, VA 22030-4444, phone (703) 764-6192.

Computer pioneer Hartmanis honored

Timothy J. Long, Ohio State University

Juris Hartmanis of Cornell University's Department of Computer Science was honored at the third annual Structure in Complexity Theory Conference in Washington, DC, June 14-17.

One of the afternoon sessions during the conference featured talks by Richard Stearns, Allan Borodin, and Paul Young reviewing Hartmanis' many contributions to computer science.

Stearns, who worked with Hartmanis at the General Electric Research Laboratories in the late 1950s and early 60s, discussed the pioneering work the two men did that marked the formal beginnings of complexity theory as a field of study. Their 1964 paper, "Computational Complexity of Recursive Sequences," gave the field its name.

Borodin, Hartmanis' first doctoral student, discussed the years after Hart-

manis left GE to establish and chair the Cornell Computer Science Department. Borodin particularly noted how influential Hartmanis has been in the overall development of computer science as a discipline, in addition to his fundamental contributions to theoretical computer science.

Young surveyed Hartmanis' more recent research concerned with isomorphism problems, sparse sets, and Kolmogorov complexity.

Hartmanis, who was marking his 60th birthday, was also honored at a banquet afterwards when 10 of his 16 graduate students recalled their years of study under Hartmanis and expressed gratitude to him for the influence he had on their lives. Phil Lewis, another Hartmanis colleague at GE, delivered a series of anecdotes about Hartmanis' career.

Keynoter discusses ways technology can help meet health care needs

Margaret Peterson, University of Connecticut Health Center

Wilbur Gantz, president of the Baxter Corporation, discussed how technology can help meet health care market needs when he delivered one of the three keynote addresses at the first Symposium on the Engineering of Computer-Based Medical Systems.

The IEEE Computer Society, the IEEE Engineering in Medicine and Biology Society, and the University of Minnesota sponsored the June 8-10 Minneapolis event, dubbed WorldMed 88. John M. Long of the University of Minnesota and chair of the Computer Society's Technical Committee on Computational Medicine was symposium chair.

In his talk, Gantz stated that intervention in chronic and fatal illness and a focus on the causes of diseases with the aid of high technology has led to genuine improvements in health care. He added that he felt costs would con-

tinue to climb but hoped this would lead to a concentration on what he termed "real" needs. Gantz stressed there is real current growth—about 7 to 8 percent—in the health care market.

John Villforth, director of the Center for Medical Devices and Radiological Health of the US Food and Drug Administration, and Werner Marly, a director of Siemens in West Germany, also presented keynote talks during WorldMed 88.

Villforth gave an amusing yet informative talk on the background and history of regulation that helped explain some confusing federal definitions and categories. Marly discussed strategic planning and the world market. His remarks demonstrated insight into and sensitivity for worldwide human needs.

Roger Schneider of the FDA opened the symposium's first technical session with a talk on the history of regulation

in the United States from 1820 to the present, beginning with the foundation of the US Pharmacopoeia established by trade and professional leaders for drug standards. At one point, Schneider stated that "software is a medical contrivance."

The proceedings of the symposium can be obtained by specifying order No. FJ863 when contacting the Computer Society Order Department, Terminal Annex PO Box 4699, Los Angeles CA 90051, phone (800) CS-BOOKS [in California, dial (714) 821-8380]. The price is \$25 if you're a Computer Society member and \$50 if you're not.

The next symposium in the series is scheduled June 26-27, 1989, also in Minneapolis. Contact the program chair, Timothy Kriewall, 3M Sarns, 6200 Jackson Road, Ann Arbor, MI 48103, for details.

State Sponsored Chair and Professor in Computer Science

NJIT invites nominations for and applications from distinguished researchers for the New Jersey State Sponsored Chair in Computer Science. The candidate will be expected to provide strong research leadership and pivotal work in any areas of integrated systems such as software engineering, distributed knowledge base systems, computer communications and networking, database systems and languages, and distributed computing. Candidates should be nationally and internationally recognized for their contributions and should have proven academic/industrial experience. He/she can expect a competitive salary for a distinguished professorship plus a research budget commensurate with responsibilities.

Computer and Information Science is a rapidly expanding department with the largest program in NJ, and offers B.S., M.S., and Ph.D. programs. CIS has extensive laboratory facilities offering excellent potential for industrial support.

NJIT is the public technological university of New Jersey, with nearly 8,000 students enrolled in baccalaureate through doctoral programs in four colleges: Newark College of Engineering, the School of Architecture, the School of Industrial Management and College of Science and Liberal Arts.

NJIT does not discriminate on the basis of sex, race, color, religion, national or ethnic origin, or age in employment.

Send resume: Personnel Box CIS.



**New Jersey
Institute of Technology**
Newark, New Jersey 07102

Visiting Professorships & Visiting Research Professorships

School of Engineering and Applied Science
The George Washington University

Visiting Professorships at junior and senior levels are available in specialty areas that include, but are not limited to, the following:

- Application Areas of Management
- Artificial Intelligence
- Cognitive Modeling
- Communications
- Computer-Aided Design
- Computer-Aided Manufacturing
- Computer-Assisted Engineering
- Computer-Integrated Manufacturing
- Computer Science
- Environmental Engineering
- Human Factors/ Man-Made Interface
- Information Management
- Management of Research and Development
- Mathematical Optimization
- Mechanical Engineering
- Reliability
- Robotics
- Simulation
- Stochastic Processes
- Structural Engineering
- Systems Analysis and Management
- Water Resources

Appointments are for one-year periods. Applicants should send vita, including complete publication list, and three references by January 16, 1989 to:

**Office of the Dean
School of Engineering and Applied Science
THE GEORGE WASHINGTON UNIVERSITY
Washington, D.C. 20052, USA**

The University is an affirmative action and equal opportunity employer.

**CALL
FOR
PAPERS,
TUTORIALS,
AND VENDOR
PROPOSALS**

**CONFERENCE ON
SOFTWARE MAINTENANCE—1989**

**PENSACOLA, FLORIDA
SEPTEMBER 1989**

The Conference on Software Maintenance—1989 (CSM-89) will gather software managers, developers, maintainers, and researchers to discuss new solutions to the continuing challenge of software maintenance and software maintainability.

Closing The Gap Between Technology and Practice

CSM-89 will focus on the processes that impact software maintenance. CSM-89 seeks papers which clearly indicate advances in the field of software maintenance, the time frame for achieving those advances; and any evidence suggesting that the advances can in fact be realized. Papers which indicate paths that should not be followed, along with evidence supporting these conclusions, are also solicited.

SPONSORS:



The IEEE Computer Society
Technical Committee on Software Engineering*



The Institute of Electrical and Electronics Engineers, Inc.



National Institute of Science and Technology (NIST)
(formerly National Bureau of Standards)

IN COOPERATION WITH:



ACM/SIGSOFT—
Special Interest Group on Software Engineering*



Association for Women in Computing (AWC)*



Software Maintenance Association (SMA)*

*Previous sponsors CSM-89 sponsorship requested

SUGGESTED TOPICS

Paper and panel session proposals related to, but not limited to, the following topics are invited

- Software Maintenance Environments
- Empirical Maintenance Studies
- Expert Systems and Software Maintenance
- Software Reusability in Maintenance
- Configuration Management
- Developing Maintainable Systems
- Standards for Software Maintenance
- Software Maintenance Metrics
- Maintenance of Ada ~ Programs
- Software Tools
- CASE
- Maintenance of Distributed Systems
- Maintenance of Fourth Generation Programs
- Software Maintenance Education
- Acquisition of Software Maintenance Services
- Software Testing
- Impact of PDLs on Software Maintainability
- Restructuring/Reengineering

* trademark of the U.S. Department of Defense

SCOPE AND PURPOSE

Software Maintenance is the enhancement, restructuring and correction of software in production use. CSM-89 will provide a forum for discussing software maintenance, distilling current experiences, and highlighting promising approaches, through presentations of refereed papers, panel sessions, tutorials, informal meetings, and tool demonstrations. CSM-89 will acquaint managers and practitioners with current advances and researchers with current needs. The conference is open to all who are involved with or have an interest in software maintenance: professionals and researchers, from industry, Government, and academia.

IMPORTANT DATES:

Submission Deadline: February 15, 1989

Acceptance Notification: March 15, 1989

Final Version: April 30, 1989

SUBMISSION INFORMATION

Papers: (5 copies double spaced) Papers should be 1000-5000 words in length. Papers must not have been published or submitted elsewhere for publication. The cover letter must include: title and maximum 250 word abstract only. The first page must include title, all authors' names, complete mailing addresses, and telephone numbers. If the paper is accepted, one of the authors is expected to present the paper at CSM-89.

Panel Session Proposals: (5 copies) Panel session proposals must include: name of panel session organizer, mailing address, and telephone number, panel topics, significance for CSM-89, and a list of 4-5 panelists. Panelists must have agreed to participate prior to the submission of the panel session proposal.

SUBMIT PAPERS AND PANEL SESSION PROPOSALS TO CAPTAIN THOMAS PIGOSKI AT HIS ADDRESS BELOW

Vendor Proposals and Tutorial Lectures: Send for information from Wilma Osborne at her address below

CONFERENCE COMMITTEE

GENERAL CHAIR

Wilma Osborne
National Bureau of Standards
Bldg 225, Rm B266
Gaithersburg, MD 20899
(301) 975-3339

PROGRAM CHAIR:

Captain Thomas Pigoski
NSGO Pensacola
Corey Station
Pensacola, FL 32511
(804) 452-6399

IMMEDIATE PAST

GENERAL CHAIR

Robert Arnold
Software Productivity Consortium (SPC)
1880 Campus Commons Drive, North
Reston, VA 22091
(703) 391-1898

CALENDAR

November 1988

SIGSoft 88, Third Symp. on Software Development Environments, Nov. 28-30, Boston. Sponsor: ACM. Contact Barry Boehm, TRW-DSG, 1 Space Park, RD-2086, Redondo Beach, CA 90278.

Micro 21, Workshop on Microprogramming and Microarchitecture, Nov. 28-Dec. 1, San Diego, Calif. Cosponsor: ACM. Contact Wen-Mei Hwu, Coordinated Science Lab, Univ. of Illinois at Urbana-Champaign, 1101 W. Springfield Ave., Urbana, IL 61801, phone (217) 244-8270.

Conf. on Neural Information Processing Systems, Nov. 28-Dec. 1, Denver. Sponsor: IEEE. Contact Scott Kirkpatrick, IBM T.J. Watson Research Center, PO Box 704, Yorktown Heights, NY 10598.

FGCS 88, Int'l Conf. on Fifth-Generation Computer Systems, Nov. 28-Dec. 2, Tokyo. Cosponsors: Inst. for New Generation Computer Technology, et al. Contact Hideo Aiso, Electrical Engineering Dept., Keio Univ., 3-14-1, Hiyoshi, Kohoku-ku, Yokohama, Kanagawa, 223 Japan, phone 81 (44) 63-1141, ext. 3320.

December 1988

Globecon 88, Dec. 1-2, Hollywood, Fla. Sponsor: IEEE. Contact Dennis J. Sassa, Bell Communications Research, NVC 22-129, 331 Newman Springs Rd., Red Bank, NJ 07701-7020, phone (201) 446-6787.

Int'l Symp. on Databases for Parallel and Distributed Systems, Dec. 5-7, Austin, Texas. Cosponsors: ACM. Contact Joseph E. Urban, ECE Dept., Univ. of Miami, PO Box 248294, Coral Gables, FL 33124, phone (305) 284-3598; or Won Kim, MCC, 3500 Balcones Center Dr., Austin, TX 78759, phone (512) 338-3439.

Third Decision Support Technology Conf., Dec. 5-7, Cambridge, Mass. Contact Donna Kacin, 51 Church St., Boston, MA 02116, phone (617) 482-3596.

ICCV 88, Second Int'l Conf. on Computer Vision, Dec. 5-8, Tampa, Fla. Contact Ruzena Bajcsy, Computer and Information Science Dept., Univ. of Pennsylvania, 200 S. 33rd St., Philadelphia, PA 19104-6389, phone (215) 898-6222.

Ninth Real-Time Systems Symp., Dec. 6-8, Huntsville, Ala. Contact Walter L. Heimerding, Honeywell Systems and

Conferences that the IEEE Computer Society sponsors or participates in are indicated by the Computer Society logo; additional conference sponsors are also listed. Other conferences of interest to our readers are also included.

For inclusion in Call for Papers or Calendar, submit information six weeks before the month of publication (i.e., for the January 1989 issue, send information for receipt by Nov. 15, 1988) to Chuck Governale, Calendar Dept., Computer, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720.

Research Center, MN65-2100, 3660 Technology Dr., Minneapolis, MN 55418, phone (612) 782-7319.

Infomatics 88, Dec. 6-8, Hong Kong. Sponsor: Int'l Information Management Congress. Contact IIMC, 345 Woodcliff Dr., Fairport, NY 14450, phone (716) 383-8330.

EuroComm 88, Second Int'l Exhibition and Congress on Business, Public, and Home Communications, Dec. 6-9, Amsterdam. Contact H. Heringa, EuroComm 88, c/o RAI Gebouw by, Europaplein, 1078 GZ Amsterdam, The Netherlands, phone 31 (20) 549-1212.

Communications into the 90s, Dec. 7-8, Ottawa, Canada. Sponsor: Carleton Univ. Contact Marg Coll, 1138 Sherman Dr., Ottawa, Ont., Canada, K2C 2M4, phone (613) 224-1741.

Int'l Seminar on Performance of Distributed and Parallel Systems, Dec. 7-9, Kyoto, Japan. Contact Yutaka Takahashi, Applied Mathematics and Physics Dept., Kyoto Univ., Kyoto 606, Japan, phone 81 (75) 751-2111, ext. 5513.

Fourth National Convention of Computer Engineers, Dec. 9-11, Calcutta, India. Sponsor: Institution of Engineers (India). Contact Anirban Basu, West Bengal State Center, 8, Gokhale Rd., Calcutta 700 020, India, phone 91 (33) 28-8914.

Winter Simulation Conf., Dec. 12-14, San Diego, Calif. Cosponsors: Society for Computer Simulation, et al. Contact Peter Haigh, NCR, 1700 S. Patterson Blvd., SER Bldg., Dayton, OH 45479, phone (513) 865-8042.

Third IEEE Conf. in Electromagnetic Field Computation, Dec. 12-14, Bethesda, Md. Contact K. Webb, Electrical Engineering Dept., Univ. of Maryland, College Park, MD 20742.

Fourth Aerospace Computer Security Applications Conf., Dec. 12-16, Orlando, Fla. Cosponsors: American Society for Information Science, Aerospace Computer Security Associates. Contact Marshall

Abrams, 1820 Dolley Madison Blvd., McLean, VA 22101, phone (703) 883-6938; or Robert D. Kovach, ACSA, c/o ORI, 1375 Piccard Dr., Rockville, MD 20850, phone (703) 648-0686.

Second Int'l Workshop of VLSI Design, Dec. 15-18, Bangalore, India. Sponsor: Computer Society of India. Contact Ravi M. Apte, Valid Logic Systems, 2820 Orchard Pkwy., San Jose, CA 95134, phone (408) 432-9400; or A. Prabhakar, Indian Telephone Industries, Dooravani Nagar, Bangalore 560 016, India, phone 91 (812) 563-2111.

Int'l Computer Science Conf., Dec. 19-21, Hong Kong. Contact Kam Wing Ng, Computer Science Dept., Chinese Univ. of Hong Kong, Shatin, N.T., Hong Kong; Jean-Louis Lassez, IBM T.J. Watson Research Center, PO Box 218, Yorktown Heights, NY 10598; or Francis Y.L. Chin, Center of Computer Studies and Application, Univ. of Hong Kong, Hong Kong.

ICS 88, Int'l Computer Symp., Dec. 20-21, Taipei, Taiwan. Contact Louis R. Chow, Tamkang Univ., Tamsui, Taiwan, Republic of China; or Shi-Kuo Chang, Computer Science Dept., Univ. of Pittsburgh, Pittsburgh, PA 15260.

Eighth Conf. on Foundations of Software Technology and Theoretical Computer Science, Dec. 21-23, Pune, India. Cosponsors: Tata Inst. of Fundamental Research, et al. Contact K.V. Nori, TRDDC, 1 Mangaldas Rd., Pune 411001, India, phone 91 (212) 61-608.

10th Israel Convention on CAD/CAM and Robotics, Dec. 27-29, Tel Aviv. Sponsor: Israel Society for Computer-Aided Design and Manufacturing. Contact Secretariat, Ortra, Ltd., PO Box 50432, Tel Aviv 61500, Israel, phone 972 (3) 66-48-25.

January 1989

Int'l Conf. on Wafer Scale Integration, Jan. 3-5, San Francisco. Contact Earl

Swartzlander, TRW, R2/2076, 1 Space Park, Redondo Beach, CA 90278, phone (213) 812-0791.

22nd Hawaii Int'l Conf. on Systems Sciences, Jan. 3-6, Kailua-Kona, Hawaii. Cosponsor: Univ. of Hawaii. Contact Ralph Sprague, Jr., Decision Sciences Dept., Univ. of Hawaii, 2404 Maile Way, E-303, Honolulu, HI 96822, phone (808) 948-7430.

Impact of Recent Computer Advances on Operations Research, Jan. 4-6, Williamsburg, Va. Sponsor: Operations Research Society of America. Contact Ramesh Sharda, College of Business Administration, Oklahoma State Univ., Stillwater, OK 74078, phone (405) 624-5113.

Western Multiconference, Jan. 4-6, San Diego, Calif. Sponsor: Society for Computer Simulation. Contact SCSS, PO Box 17900, San Diego, CA 92117.

Qualtech 89, Jan. 11-12, Los Angeles. Sponsor: Los Angeles Section, American Society for Quality Control. Contact Terry Mohr, Northrop Aircraft, Dept. 7420/94, 1 Northrop Ave., Hawthorne, CA 90250, phone (213) 416-2056.

Second Int'l Workshop on Artificial Intelligence in Economics and Management, Jan. 11-13, Singapore. Cosponsors: IFIP, et al. Contact Jazar Motiwalla or Vicky Toh, Inst. of Systems Science, National Univ. of Singapore, Heng Mui Keng Terrace, Singapore 0511, phone (65) 772-2075.

Symp. on Electronic Imaging, Jan. 15-20, Los Angeles. Sponsors: Int'l Society for Optical Engineering, Society for Imaging Science and Technology. Contact SPIE, PO Box 10, Bellingham, WA 98227-0010.

Saudi Computer 89, Jan. 29-Feb. 2, Riyadh, Saudi Arabia. Contact Gerald G. Kallman, 5 Maple Ct., Ridgewood, NJ 07450-4431, phone (201) 652-7070.

Unix Winter Technical Conf., Jan. 30-Feb. 3, San Diego, Calif. Sponsor: Unix Assoc. Contact Judith DesHarnais, Unix Conf. Office, PO Box 385, Sunset Beach, CA 90742, phone (213) 592-1381.

Database 89 Expo and Conf., Jan. 31-Feb. 2, San Francisco. Contact Dana De Nardi, 289 S. San Antonio Rd., No. 204, Los Altos, CA 94022, phone (415) 941-8440.

February 1989

Fifth Int'l Conf. on Data Engineering, Feb. 7-9, Los Angeles. Contact John Carls, Computer Science Dept., Univ. of Minnesota, 207 Church St., SE, Minneapolis, MN 55455, phone (612) 625-6092; Richard L. Shuey, Rensselaer Polytechnic Inst., Dept. of Computer Science, Troy, NY

12180, phone (518) 276-8376; or Data Engineering 89, IEEE Computer Society, 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

1989 Aerospace Applications Conf., Feb. 12-17, Breckenridge, Colo. Sponsor: IEEE South Bay Harbor Section. Contact Harry Endert, 15137 Gilmore St., Van Nuys, CA 91411.

17th Computer Science Conf., Feb. 21-23, Louisville, Ky. Sponsor: ACM. Contact Conf. Dept. A, Assoc. for Computing Machinery, 11 W. 42nd St., New York, NY 10036.

Comcon Spring 89, Feb. 27-Mar. 3, San Francisco. Contact Kenichi Miura, Computational Research Dept., MS B2-7, Fujitsu America, 3055 Orchard Dr., San Jose, CA 95134-2017, phone (409) 432-1300, ext. 5408 or 5723.

Securicom 89, Seventh Worldwide Congress on Computer and Communications Security and Protection, Feb. 28-Mar. 3, Paris. Contact SEDEP, 8, rue de la Michodière, 75002, Paris, France, phone 33 (1) 47-42-40-30.

March 1989

Fourth Conf. on Hypercube Concurrent Computers and Applications, Mar. 6-8, Monterey, Calif. Sponsors: US Dept. of Energy, et al. Contact John Gustafson, Div. 1413, Sandia National Labs, PO Box 5800, Albuquerque, NM 87185.

EMC 89, Eighth Zurich Symp. and Technical Exhibition on Electromagnetic Compatibility, Mar. 6-9, Zurich. Sponsor: Swiss Electrotechnical Assoc. Contact T. Dvorak, EMC 89, ETH Zentrum-IKT, CH-8092, Zurich, Switzerland, phone 41 (1) 256-2790.

CAIA 89, Fifth Int'l Conf. on Artificial Intelligence Applications, Mar. 6-10, Miami, Fla. Contact CAIA 89, IEEE Computer Society, 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

Seventh IEEE Computer Fair, Mar. 8-10, Huntsville, Ala. Joint sponsors: Huntsville IEEE Section, Huntsville IEEE Computer Society. Contact Connie Harbison, TotalCom, 1115 Church St., Suite H, Huntsville, AL, phone (205) 534-6383.

NCAT, Seventh National Conf. on Ada Technology, Mar. 13-16, Atlantic City, NJ. Contact NCAT, US Army Communications—Electronics Command, Attn.: AMSEL-RD-SE-CRM (Kay Trezza), Fort Monmouth, NJ 07703-5000, phone (201) 532-1898.

Tapsort 89, Joint Conf. on Theory and Practice of Software Development, Mar. 13-17, Barcelona, Spain. Sponsor: European Assoc. of Theoretical Computer Science. Contact

Pere Botella, Facultat d'Informàtica, Pau Gargallo, 5, 08028 Barcelona, Spain, phone 34 (3) 333-8308.

Workshop on Visual Motion, Mar. 20-22, Irvine, Calif. Contact Ellen Hildreth, Artificial Intelligence Lab, 545 Technology Square, Cambridge, MA 02139; or Ramesh Jain, Electrical Engineering and Computer Science Dept., Univ. of Michigan, Ann Arbor, MI 48109-2122.

PCCC 89, Phocnix Conf. on Computers and Communications, Mar. 22-24, Scottsdale, Ariz. Cosponsor: Arizona State Univ. Contact Thaddeus Regulinski, Loral Corp., PO Box 295, Goodyear, AZ 85338.

AISIG 89, Artificial Intelligence Systems in Government Conf., Mar. 27-31, Washington, DC. Cosponsor: George Washington Univ. Contact AISIG 89, MS W418, Mitre Corp., 7525 Colshire Dr., McLean, VA 22102; or IEEE Computer Society, 1730 Massachusetts Ave. NW, 20036-1903, phone (202) 371-1013.

Eastern Multiconference, Mar. 28-31, Tampa, Fla. Sponsor: Society for Computer Simulation. Contact SCSS, PO Box 17900, San Diego, CA 92117-7900, phone (619) 277-3888.

Third Parallel Processing Symp., Mar. 29-31, Fullerton, Calif. Cosponsors: California State Univ. at Fullerton, et al. Contact Larry Canter, 1619 N. Hale, Fullerton, CA 92631, phone (714) 738-3414.

Eighth Symp. on Principles of Database Systems, Mar. 29-31, Philadelphia. Sponsor: ACM. Contact Avi Silbershatz, Computer Science Dept., Univ. of Texas at Austin, Austin, TX 78712.

Workshop on Applied Computing 89, Mar. 30-31, Stillwater, Okla. Cosponsors: National Science Foundation, et al. Contact Donald D. Fisher, Oklahoma State Univ., CIS, Stillwater, OK 74078, phone (405) 624-5668.

CMC 89, Colorado Microelectronics Conf., Mar. 30-31, Colorado Springs, Colo. Contact Conf. Secretary, CMC 89, Microelectronics Research Labs, Univ. of Colorado, Box 7150, Colorado Springs, CO 80933-7150, phone (719) 593-3488.

Western Educational Computing Conf., Mar. 30-31, Santa Cruz, Calif. Sponsor: California Educational Computing Consortium. Contact Judah Rosenwald, Extended Education, NAD 153, San Francisco State Univ., 1600 Holloway, San Francisco, CA 94132, phone (415) 338-1212.

April 1989

Second National Conf. on Telecommunications, Apr. 2-5, York, England. Sponsor:

Institution of Electrical Engineers. Contact IEE Conf. Services, Savoy Pl., London WC2R 0BL, UK, phone 44 (1) 240-1871, ext. 222.

ASPLOS III, Third Int'l Conf. on Architectural Support for Programming Languages and Operating Systems, Apr. 3-6, Boston. Cosponsors: ACM. Contact Joel Emer, DEC/MIT, 545 Technology Square (NE43-503), Cambridge, MA 02139.

Working Conf. on Visual Database Systems, Apr. 3-7, Tokyo. Cosponsors: IFIP, IPSJ. Contact IFIP TC-2 Working Conf., Tosiya L. Kunii, Information Science Dept., Faculty of Science, Univ. of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113, Japan, phone 81 (3) 812-2111, ext. 4116.

Third Eurographics Workshop on Intelligent CAD Systems, Apr. 3-7, Texel, The Netherlands. Sponsor: Center for Mathematics and Computer Science. Contact Marja Hegt, CMCS, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands, phone 31 (20) 592-4058.

Int'l Symp. on Database Systems for Advanced Applications, Apr. 10-12, Seoul, Korea. Cosponsors: IPSJ, Korean Information Science Society. Contact Sukho Lee, Computer Engineering Dept., Seoul National Univ., Sinlim-Dong, Gwanak-ku, Seoul 151, Korea, phone 82 (2) 886-0101.

MIV 89, Int'l Workshop on Industrial Applications of Machine Intelligence and Vision,

Apr. 10-12, Tokyo. Cosponsors: IEEE, et al. Contact Mitsuru Ishizuka, Inst. of Industrial Science, Univ. of Tokyo, 7-22-1, Roppongi, Minato-ku, Tokyo, 106, Japan, phone 81 (03) 470-5389.

1989 IEEE VLSI Test Workshop, Apr. 11-13, Atlantic City, NJ. Sponsors: IEEE Computer Society Test Technology Committee, IEEE Philadelphia Section. Contact Wesley E. Radcliffe, IBM East Fishkill, Dept. 277, Bldg. 321-5E1, Hopewell Junction, NY 12533.

ETC 89, First European Test Conf., Apr. 12-14, Paris. Cosponsor: Societe des Electriciens et des Electroniciens. Contact Colin Maunder, British Telecom Research Labs, Martlesham Heath, Ipswich, Suffolk IP5 7RE, phone 44 (473) 642-706.

First Symp. on Parallel and Distributed Processing, Apr. 20-21, Dallas. Contact Behrooz Shirazi, Computer Science Dept., Southern Methodist Univ., Dallas, TX 75275, phone (214) 692-2874.

Multimedia 89, Second Cosmoc Int'l Multimedia Communications Workshop, Apr. 20-23, Ottawa, Canada. Cosponsors: IEEE, et al. Contact Ottawa Carleton Research Inst., 1150 Morrison Dr., Suite 302, Ottawa, Ont., Canada K2H 8S9, phone (613) 726-8827.

Infocom 89, Conf. on Computer Communications, Apr. 24-27, Ottawa,

Canada. Contact Celia Desmond, Telecom Canada, 438 Bay St., Floor 5, South Tower, Toronto, Ont., Canada, M5G 2E1, phone (416) 581-2318.

Vision 89, Apr. 25-27, Chicago. Sponsor: Society of Manufacturing Engineers. Contact Maria Nowakowski, SME, 1 SME Dr., PO Box 930, Dearborn, MI 48121, phone (313) 271-1500, ext. 376.

CHI 89, Conf. on Human Factors in Computing Systems, Apr. 30-May 4, Austin, Texas. Cosponsors: ACM, Human Factors Society. Contact Claudia Raun or Bill Curtis, MCC, PO Box 200195, Austin, TX 78720, phone (512) 338-3798.

34th Int'l Instrumentation Symp., Apr. 30-May 4, Orlando, Fla. Sponsor: Instrument Society of America. Contact Frederick A. Kern, PO Box 65, Seaford, VA 23696, phone (804) 865-3269.

May 1989

1989 IEEE Symp. on Research in Security and Privacy, May 1-3, Oakland, Calif. Contact Terry V. Benz, Trusted Information Systems, 11340 Olympic Blvd., Suite 265, Los Angeles, CA 90064, phone (213) 477-5828.

IEEE TRANSACTIONS ON COMPUTERS

Setting the standards for computing in the '80s, the '90s, and into the next century.

IEEE Transactions on Computers publishes thoroughly researched and refereed papers covering the theory, design, and applications of computer systems. Topics addressed in *Transactions on Computers* include

- Circuits, systems, and networks
- VLSI and digital devices
- Fault tolerant computing
- Distributed systems
- Testing and performance evaluation
- Algorithms

Three special issues each year focus on topics with far-reaching implications in computing. Special issues in 1989 will cover

- High-yield VLSI systems (April)
- Distributed Computing Systems (August)
- Computer Systems Performance (December)

Currently planned for 1990 are special issues on fault-tolerant computing systems, computer arithmetic, and supercomputing systems.

DON'T MISS A SINGLE ISSUE! If you currently subscribe to *Transactions on Computers*, now is the perfect time to send in your renewal payment for 1989. If you are not currently a subscriber, you can add *Transactions on Computers* to your 1989 membership renewal, or order your subscription by returning the coupon below or filling out the membership application elsewhere in this magazine.

SIGN ME UP! Enter my 1-year subscription to *IEEE Transactions on Computers!*

- ☐ I am a member of the IEEE Computer Society (paid through 1989), so I am enclosing the member price of just \$20.
- ☐ I am NOT a member of the IEEE Computer Society, but I am a member of IEEE, ACM, SCS, DPMA, or another technical professional society, so I am enclosing the sister society price of just \$36.

Signature _____

Mail to: IEEE Computer Society
Circulation Department
10662 Los Vaqueros Circle
Los Alamitos, CA 90720

NAME _____

ADDRESS _____

CITY/STATE/ZIP _____

COUNTRY _____

MEMBER OF _____

MEMBER NUMBER _____

☐ Check IN \$US payable to IEEE enclosed.
Charge my ☐ Visa ☐ MasterCard ☐ American Express

Account No. _____ Expires _____

Third AI Research in Environmental Science Workshop, May 2-4, Washington, DC. Contact William R. Moninger, Environmental Research Labs, NOAA, R/E2, 325 Broadway, Boulder, CO 80803, phone (303) 497-6435.

Sixth Canadian Symp. on Instructional Technology, May 3-5, Halifax, N.S., Canada. Sponsor: National Research Council Canada. Contact Laurie Forget, CSIT, Conf. Services, NRCC, Ottawa, Ont. K1A 0R6, Canada, phone (613) 993-9009.

20th Pittsburgh Conf. on Modeling and Simulation, May 4-5, Pittsburgh, Pa. Sponsor: Univ. of Pittsburgh. Contact William G. Vogt or Marlin H. Mickle, 348 Benedum Engineering Hall, Univ. of Pittsburgh, Pittsburgh, PA 15261.

STA 5, Fifth Structured Techniques Assoc. Conf., May 8-11, Chicago. Sponsors: STA, ACM. Contact STA, c/o Robert Binder Systems Consulting, Inc., 3 First National Plaza, Suite 1400, Chicago, IL 60602.

CompEuro 89, Int'l Conf. on VLSI and Computer Peripherals, May 8-12, Hamburg. Cosponsors: Gesellschaft für Informatik, et al. Contact Walter E. Proebster, IBM Lab, PO Box 1380, D-7030 Boebblingen, Federal Republic of Germany, phone 49 (70) 3116-3929.

ICCAL 89, Second Int'l Conf. on Computer-Assisted Learning, May 9-11, Dallas. Sponsor: Computer Learning Research Center, Univ. of Texas at Dallas. Contact Janet Harris, Center for Continuing Education, Univ. of Texas at Dallas, PO Box 830688, MS CN 1.1, Richardson, TX 75083-0688.

Robots 13, May 9-12, Gaithersburg, Md. Sponsor: SME. Contact Rebecca Alsop, SME, 1 SME Dr., Dearborn, MI 48121, phone (313) 271-1500, ext. 358.

CCC 89, Second Hungarian Custom Circuit Conf., May 10-12, Szeged, Hungary. Cosponsors: Scientific Society of Measurement and Automation (MATE). Contact MATE Secretariat, 1055 Budapest, Kossuth L. tér 6-8, Hungary, phone (1) 531-406.

Sixth IEEE Workshop on Real-Time Operating Systems and Software, May 11-12, Pittsburgh. Cosponsor: Carnegie Mellon Univ. Contact Andre van Tilborg, Office of Naval Research, 800 N. Quincy St., Arlington, VA 22217-5000, phone (202) 696-4302.

AAMSI Congress 89, May 11-13, San Francisco. Sponsor: American Assoc. for Medical Systems and Informatics. Contact AAMSI, Suite 700, 1101 Connecticut Ave. NW, Washington, DC 20036, phone (202) 857-1189.

Int'l Conf. on Robotics and Automation, May 14-19, Scottsdale, Ariz. Sponsor: IEEE. Contact Harry Hayman, PO Box 3216, Silver

Spring, MD 20901, phone (301) 434-1990 or (407) 483-3037.

11th Int'l Conf. on Software Engineering, May 15-18, Pittsburgh. Cosponsor: ACM. Contact Larry Druffel, Software Engineering Inst., Carnegie Mellon Univ., Pittsburgh, PA 15213-3890, phone (412) 268-7740.

Sixth Conf. on Real-Time Applications in Nuclear, Particle, and Plasma Physics, May 15-18, Williamsburg, Va. Sponsors: IEEE, et al. Contact Roy Whitney, 12000 Jefferson Ave. Newport News, VA 23606, phone (804) 249-7536.

18th Mumps Users Group Annual Meeting, May 15-19, Seattle. Contact Mumps Users Group, 4321 Hartwick Rd., Suite 510, College Park, MD 20740, phone (301) 779-6555.

SID 89, Society for Information Display Int'l Symp., Seminar, and Exhibition, May 15-19, Baltimore. Contact Society for Information Display, c/o Palisades Inst. for Research Services, 201 Varick St., Rm. 1140, New York, NY 10014, phone (212) 620-3375.

Int'l Symp. on VLSI Technology, Systems and Applications, May 17-19, Taipei, Taiwan. Cosponsors: Republic of China National Science Council, Industrial Technology Research Inst. Contact Alice M. Chiang, MIT, Lincoln Lab, Lexington, MA 02173-0073, phone (617) 981-4629.

Fifth Int'l Workshop on Software Specification and Design, May 19-20, Pittsburgh. Cosponsor: ACM. Contact Sol J. Greenspan, GTE Labs, 40 Sylvan Rd., Waltham, MA 02254, phone (617) 466-2962; or Colin Potts, MCC, 9390 Research Blvd., Kaleido II Bldg., Austin, TX 78759, phone (512) 338-3629.

First IEEE Symp. on Parallel and Distributed Processing, May 22-23, Dallas. Sponsor: Dallas Section of the IEEE Computer Society. Contact Mark Shadovs, Information Technologies Lab, Texas Instruments, PO Box 655474, MS 238, Dallas, TX 75265, or call Behrooz Shirazi at (214) 692-2874.

AMAST, Int'l Conf. on Algebraic Methodology and Software Technology, May 22-24, Iowa City, Iowa. Contact Eugene Madison or Teodor Rus, Computer Science and Mathematics Dept., Univ. of Iowa, Iowa City, IA 52242, phone (319) 335-0742 or 0694.

Sixth Int'l Conf. on Testing Computer Software, May 22-25, Washington, DC. Sponsors: DPMA, ACM. Contact Genevieve Houston-Ludlam, Frontier Technologies, 2444 Solomons Island Rd., Suite 205, Annapolis, MD 21401, phone (301) 266-8244.

10th Tunisian French Seminar of Computer Science: The Role of Languages in Programming, May 23-25, Tunis, Tunisia. Sponsor: Tunisian Information Processing Society.

Contact Ali Milli, Faculty of Sciences, Univ. of Tunis II, Campus Universitaire, 1002 Belvedere, Tunisia.

SIGMetrics 89 and Performance 89, May 23-26, Berkeley, Calif. Sponsors: ACM, IFIP. Contact Luis-Felipe Cabrera, IBM Almaden Research Center, Mail Code K52/803, San Jose, CA 95120-6099, phone (408) 927-1838.

ICCI 89, Int'l Conf. on Computing and Information, May 23-27, Toronto. Contact Waldemar W. Koczkodaj, Laurentian Univ., CoSc, Sudbury, Ont., Canada P3E 2C6, phone (705) 675-1151.

Workshop on New Directions in Computer Chess, May 28-31, Edmonton, Alta., Canada. Sponsors: Int'l Computer Chess Assoc., Canadian Information Processing Society. Contact Tony Marsland, Computing Science Dept., Univ. of Alberta, Edmonton, Alta., Canada T6G 2H1.

16th Int'l Symp. on Computer Architecture, May 28-31, Jerusalem, Israel. Cosponsor: ACM. Contact Int'l Symp. on Computer Architecture, 90A Hayarkon St., PO Box 3190, Tel Aviv 61031, Israel, phone 972 (3) 246-261.

19th Int'l Symp. on Multiple-Valued Logic, May 29-31, Guangzhou, China. Contact David M. Miller, Computer Science Dept., Univ. of Victoria, Canada, V8W 2Y2, phone (604) 721-7220.

June 1989

IEEE Pacific Rim Conf. on Communications, Computers, and Signal Processing, June 1-2, Victoria, B.C., Canada. Sponsor: IEEE Victoria Section, Univ. of Victoria. Contact Warren D. Little, Dept. of Electrical and Computer Engineering, Univ. of Victoria, PO Box 1700, Victoria, B.C., Canada V8W 2Y2, phone (604) 721-8686.

CVPR 89, Conf. on Computer Vision and Pattern Recognition, June 4-8, San Diego, Calif. Contact Rama Chellappa, PHE324, Electrical Engineering-Systems Dept., Univ. of Southern California, University Park, MC-0272, CA 90089, phone (213) 743-8559; or CVPR 89, IEEE Computer Society, 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

Fourth Israel Conf. on Computer Systems and Software Engineering, June 5-6, Tel Aviv. Cosponsors: Israel Chapter of IEEE Computer Society, Information Processing Assoc. of Israel. Contact S. Koenig, Ortra Ltd., PO Box 50432, Tel Aviv 61500, Israel, phone 972 (3) 664-825.

Fourth Symp. on Logic in Computer Science, June 5-8, Pacific Grove, Calif. Cosponsor: ACM. Contact Albert Meyer, Lab for Computer Science, MIT, 545 Technology Square, Cambridge, MA 02139.

① Ninth Int'l Conf. on Distributed Computing Systems, June 5-9, Newport Beach, Calif. Contact Kane Kim, Computer Engineering Program, Electrical Engineering Dept., Univ. of California, Irvine, CA 92717, phone (714) 856-5542.

① Working Conf. on Computer-Aided Design Systems Using Artificial Intelligence Techniques, June 6-7, Tokyo. Sponsor: IFIP, IPSJ. Contact Akihiko Yamada, NEC, 4-14-22 Shibaura, Minato-ku, Tokyo 108, Japan; or Kozo Kinoshita, Faculty of Integrated Arts and Sciences, Hiroshima Univ., 1-1-8 Sendamachi, Naka-ku, Hiroshima 730, Japan, phone 81 (82) 249-9150.

Third Int'l Workshop on Wafer Scale Integration, June 6-8, Como, Italy. Sponsor: IFIP. Contact Mariagiovanna Sami, Dip. di Elettronica, Politecnico di Milano, Piazza Leonardo da Vinci 32, I-20133 Milan, Italy, phone 39 (2) 23-99-35-16.

① Second Int'l Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, June 6-9, Tullahoma, Tenn. Sponsors: ACM, Univ. of Tennessee Space Inst. Contact Moonis Ali, Univ. of Tennessee Space Inst., Tullahoma, TN 37388, phone (615) 455-0631.

Workshop on Automatic Verification Methods for Finite State Systems, June 12-14, Grenoble, France. Sponsor: C-cube, the French National Project on Parallelism. Contact Edmund M. Clarke, Jr., Carnegie Mellon Univ., Computer Science Dept., Pittsburgh, PA 15213; A. Pnueli, Weizmann Inst., Rehovot, Israel; or J. Sifakis, LGI-IMAG, BP 53X, 38041 Grenoble Cedex, France.

ICAIL 89, Second Int'l Conf. on Artificial Intelligence and Law, June 13-16, Vancouver, Canada. Contact Robert T. Franssen or Joseph C. Smith, Faculty of Law, Univ. of British Columbia, Vancouver, B.C., Canada, phone (604) 228-2323.

Int'l Conf. on Software Engineering and Knowledge Engineering, June 15-17, Chicago. Contact Shi-Kuo Chang, Computer Science Dept., Univ. of Pittsburgh, 322 Alumni Hall, Pittsburgh, PA 15260, phone (412) 624-8490.

① Int'l Workshop on Hardware Fault Tolerance in Multiprocessors, June 19-20, Urbana, Ill. Contact Prith Banerjee, Coordinated Science Lab, Univ. of Illinois, 1101 W. Springfield Ave., Urbana, IL, phone (217) 333-6564.

① CHDL 89, Ninth Int'l Symp. on Computer Hardware Description Languages and Applications, June 19-21, Washington, DC. Cosponsors: IFIP, ACM. Contact John A. Darringer, IBM T.J. Watson Research Center, PO Box 218, Yorktown Heights, NY 10598, phone (914) 945-1018.

① FTCS 19, Int'l Symp. on Fault-Tolerant Computing, June 21-23, Chicago. Contact S.M. Reddy, FTCS 19, Electrical and Computer Engineering Dept., Univ. of Iowa, Iowa City, Iowa 52242, phone (319) 335-5196; or Ravi K. Iyer, Coordinated Science Lab, Univ. of Illinois, 1101 W. Springfield Ave., Urbana, IL, phone (217) 333-9732.

① CAR 89, Computer-Assisted Radiology Conf., June 25-28, Berlin. Cosponsor: AMK Berlin. Contact Michael L. Rhodes, MPDI, 2730 Pacific Coast Hwy., Torrance, CA 90505, phone (213) 539-5944.

① DAC 89, 26th Design Automation Conf., June 25-29, Las Vegas. Cosponsor: ACM. Contact Michael J. Lorenzetti, MCNC, PO Box 12889, Research Triangle Park, NC 27709-2889; or Pat Pistilli, MP Associates, 7490 Clubhouse Rd., Suite 102, Boulder, CO 80301, phone (303) 530-4333.

Research and Development

EDS, the world's leader in computer services, is seeking software research scientists to join an established Research and Development organization in Auburn Hills, Michigan. Our mission is to expand the business opportunities of EDS in the information services industry through the development and innovative application of technology. Applicants should have a Master's Degree or Ph.D. in computer science, applied mathematics, or a related technical area and interest in:

- ▶ Artificial Intelligence
- ▶ Logic Programming
- ▶ Productivity Tools for Software Engineering
- ▶ Parallel Processing
- ▶ Distributed Computing
- ▶ Animation and Graphics
- ▶ Human-Computer Interaction

Our activities are supported by networked SUN, Xerox, and Macintosh workstations with connections to large mainframe installations.

EDS Research and Development is located in suburban Oakland County, Michigan. This location affords easy access to a wide variety of cultural and recreational activities and is within commuting distance of several major universities. The school districts in Oakland County are among the best in the nation.

Send your resume in confidence to:

EDS Research and Development
Recruiting Coordinator
3551 Hamlin Road
Auburn Hills, MI 48057

Principals Only: An Equal Opportunity Employer M/F/V/H

EDS

Faculty Positions Computer and Information Science

New Jersey Institute of Technology seeks assistant, associate and full professors for spring/fall 1989. Ph.D. in computer science or closely related field required. Senior level applicants must have proven research and funding record. Positions available in, but not limited to: distributed computing including computer architecture, data communications networking, realtime computing and fault tolerance; software development including artificial intelligence, expert systems, computer graphics, office automation, data management systems, cognitive science, and computational linguistics. Department offers B.S., B.A., M.S., and Ph.D., in computer science, and Ph.D. in management jointly with Rutgers-Newark. Computing facilities include VAX 8800, VAX 8530, IBM 4361, SUN workstations, Symbolics machines, TI Explorers and graphics systems.

NJIT is the comprehensive technological university of New Jersey with nearly 8,000 students enrolled in Newark College of Engineering, the School of Architecture, College of Sciences and Liberal Arts and the School of Industrial Management.

NJIT does not discriminate on the basis of sex, race, color, handicap, national or ethnic origin, or age in employment.

Send resume and names of at least three references to: Personnel Box C15

NJIT

**New Jersey
Institute of Technology**
Newark, New Jersey 07102

CALL FOR PAPERS and Panel Session Proposals

The Thirteenth Annual International
**Computer Software &
Applications Conference**

compsac89



Hyatt Orlando Hotel
Orlando, Florida

TUTORIALS: September 18-19, 1989 **CONFERENCE:** September 20-22, 1989

DEADLINES:

- February 1, 1989 for receiving all papers and proposals
- March 3, 1989 panel organizers notified of acceptance
- March 22, 1989 organizers of accepted panel proposals provide final information on session chairmen and panelists
- April 7, 1989 authors notified of acceptance
- June 1, 1989 camera-ready copies of accepted papers and panelists' position papers due.

INFORMATION FOR AUTHORS:

- 5 copies of the full paper (double-spaced) 1000-5000 words
- Proposals should include: title, organizer/affiliations/address/phone number and 150-word abstract.

INFORMATION FOR PANEL ORGANIZERS:

- 5 copies of the panel proposal
- Proposals should include: title, organizer/affiliations/address/phone number and a 150-word scope statement.

Papers and Panel Session Proposals related to, but not limited to, the following areas are invited:

Application Software and Experience in the areas of:

- Integrated Office Automation Systems
- Integrated Communication Networks
- Integrated Engineering Workstations
- Image Communication
- CAD/CAM/CIM

Software Quality and Productivity

- Software Configuration Management
- Metrics & Measurements
- Quality Assurance
- Productivity
- Standards

Software Engineering Management

- New Management Approaches
- Software Factory Concepts
- Database/Distributed Processing Applications

Development and Maintenance Environments

- Industrial Environment
- Tools and Methodology Support
- Knowledge Based Techniques

Object-Oriented Computing

- Object-oriented Programming Languages
- Object-oriented Database Management Systems
- Architectural Support for Object-oriented Computing

Expert/Knowledge Based Systems

- Integrated Knowledge Based Systems
- Knowledge Base Computers
- Practical Applications

Ultra-Reliable Real-Time Systems

- Distributed Real-Time Systems
- Fault Tolerant Software Systems

Emerging Technologies

- Heterogeneous Information Systems
- Intelligent Human Interface
- Software for Massively Parallel Systems
- Neural Networks
- Optical Computing

Submit papers and panel proposals to:

Professor Stanley Y.W. Su
Program Chair, COMPSAC89
301 Computer Science &
Engineering Building
Database Systems Research &
Development Center
University of Florida
Gainesville, Florida 32611 U.S.A.
Phone: 904-335-8458 or 904-335-8460
Netmail: su@ufl.edu
Telefax: 1-904-335-8008

Conference Chairman

Mr. Jack Cline
Vice President
Martin Marietta Data Systems
Mail Point 165
P. O. Box 590385
Orlando, Florida 32859-0385 U.S.A.



IEEE COMPUTER SOCIETY



THE INSTITUTE OF ELECTRICAL
AND ELECTRONICS ENGINEERS, INC.

CALL FOR PAPERS

Call for papers and referees for Computer



Computer seeks articles for inclusion in an upcoming special issue.

Visualization in Scientific Computing has been selected as the theme for August 1989. Tutorial, survey, descriptive, case-study, applications-oriented, or pedagogic manuscripts are sought.

Suggested topics include

- hardware strategies for scientific visualization; parallel architectures, supercomputing, workstations, and networks
- user interfaces, paradigms and interactive techniques for visualization of 3D (and higher dimensional) models
- scientific data analysis, manipulation, representation, and display techniques, particularly volume visualization and methods for large multivariate data sets
- dissemination of the results of interactive systems and animated images; standards; and televisualization and networks, and
- application of visualization techniques

niques to science and engineering problems and data.

Papers must not have been previously published or currently submitted for publication elsewhere.

A 300-word abstract is due as soon as possible. Eight copies of the full manuscript must be submitted by Dec. 1, 1988, to Gregory N. Nielson, Computer Science Dept., Arizona State University, Tempe, AZ 85287-5406, phone (602) 965-2785, electronic mail address: nielson@enuxva.asu.edu.

Authors will be notified of acceptance by Mar. 1, 1989. The final version of the manuscript is due no later than May 1, 1989.

Referees: Persons interested in serving as referees are asked to send a note with a list of technical interests to Nielson or Bruce D. Shriver, *Computer Editor-in-Chief*, Department of Decision Sciences, University of Hawaii, 2404 Maile Way, Honolulu, HI 96822; E-mail address: shriver@uhccux.uhcc.hawaii.edu.

Dec. 2, 1988, to Lynne A. Henderson, Palisades Inst. For Research Services, 201 Varick St., Rm. 1140, New York, NY 10014, phone (212) 620-3375.

Seventh National Conf. on Ada Technology: Mar. 13-16, 1989, Atlantic City, NJ. Submit manuscript by Dec. 9, 1988, to Seventh National Conf. on Ada Technology, US Army Communications—Electronics Command, Attn.: AMSEL-RD-SE-CRM (Kay Trezza), Fort Monmouth, NJ 07703-5000, phone (201) 532-1898.

Int'l Conf. on Mathematics of Program Construction: June 26-30, 1989, Groningen, The Netherlands. Submit paper by Dec. 9, 1988, to Jan L.A. van de Snepscheut, Mathematics and Computing Science Dept., Groningen Univ., PO Box 800, 9700 AV Groningen, The Netherlands.

11th Int'l Joint Conf. on Artificial Intelligence: Aug. 20-26, 1989, Detroit. Sponsors: Int'l Joint Conf. on Artificial Intelligence, American Assoc. for Artificial Intelligence. Submit paper by Dec. 12, 1988, to N.S. Sridharan, Central Engineering Labs, FMC Corp., 1205 Coleman Ave., Box 580, Santa Clara, CA 95052, phone (408) 289-0315.

18th Int'l Conf. on Parallel Processing: Aug. 8-12, 1989, St. Charles, Ill. Sponsor: Pennsylvania State Univ. Submit paper by Dec. 15, 1988, to Peter M. Kogge, MS 0302, IBM, Route 17C, Oswego, NY 13827, phone (607) 751-2291.

Second Int'l Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems: June 6-9, 1989, Tullahoma, Tenn. Sponsors: ACM, Univ. of Tennessee Space Inst. Submit abstract by Dec. 15, 1988, to Moonis Ali, Univ. of Tennessee Space Inst., Tullahoma, TN 37388, phone (615) 455-0631.

First Conf. of the Int'l Assoc. of Knowledge Engineers: June 26-28, 1989, College Park, Md. Sponsors: Int'l Assoc. of Knowledge Engineers, Univ. of Maryland. Submit abstract by Dec. 15, 1988, and complete paper by Mar. 31, 1989, to Michael Teague, IAKE/UM Committee for Conf. Papers, Int'l Assoc. of Knowledge Engineers, Georgetown PO Box 25461, Washington, DC 20007.

CMC 89, Colorado Microelectronics Conf.: Mar. 30-31, 1989, Colorado Springs, CO. Submit abstract by Dec. 15, 1988, Technical Chair, CMC 89, Microelectronics Research Labs, Univ. of Colorado at Colorado Springs, Box 7150, Colorado Springs, CO 80933-7150.

Systems Science X Int'l Conf.: Sept. 19-22, 1989, Wrocław, Poland. Submit abstract by



IEEE Trans. on Knowledge and Data Engineering will begin publication in March 1989. Papers are sought, particularly those emphasizing research, design, and development of knowledge and data-engineering methodologies, strategies, and systems. For information and submission requirements, contact C.V. Ramamoorthy, Computer Science Div., Univ. of California, Berkeley, CA 94720, phone (415) 642-4751; or Benjamin W. Wah, Coordinated Science Lab, Univ. of Illinois, 1101 W. Springfield Ave., Urbana, IL 61801, phone (217) 333-3516.

1989 Aerospace Applications Conf.: Feb. 12-17, 1989, Breckenridge, Colo. Sponsor: IEEE South Bay Harbor Section. Submit paper to Leo Mallette, 2309 S. Santa Anita Ave., Arcadia, CA 91006.

Eighth Australia Conf. on Microelectronics: July 12-14, 1989, Brisbane, Australia. Cosponsors: IEEE Queensland Section, Institution of Engineers, Australia. Submit synopsis to Microelectronics 89, 11 National Circuit, Barton ACT 2600, Australia, phone (61) 602-766-549.

1989 IEEE VLSI Test Workshop: Apr. 11-13, 1989, Atlantic City, NJ. Sponsor: IEEE Computer Society Test Technol-

ogy Committee, IEEE Philadelphia Section. Submit abstract by Nov. 25, 1988, to Mukund Modi, Naval Air Engineering Center, ATE Software Center, Code: 52514, Lakehurst, NJ 08733, phone (201) 323-2560.

Sixth Int'l Conf. on Testing Computer Software: May 22-25, 1989, Washington, DC. Sponsors: DPMA, ACM. Submit paper by Nov. 30, 1988, to Boris Beizer, 1232 Glenbrook Rd., Huntingdon Valley, PA 19006.

Third Int'l Workshop on Wafer Scale Integration: June 6-8, 1989, Como, Italy. Sponsor: IFIP. Submit extended abstract by Dec. 1, 1988, and full paper by Apr. 10, 1989, to Mariagiovanna Sami, Dip. di Elettronica, Politecnico di Milano, Piazza Leonardo da Vinci 32, I-20133 Milan, Italy, phone 39 (2) 23-9935-16.

35th IEEE-Holm Conf. on Electrical Contacts: Sept. 18-20, 1989, Chicago. Submit abstract by Dec. 1, 1988, to IEEE-Holm Conf., IEEE Headquarters, 345 E. 47th St., New York, NY 10017-2394, phone (212) 705-7405.

SID 89, Society for Information Display Int'l Symp., Seminar, and Exhibition: May 15-19, 1989, Baltimore. Submit paper by

Dec. 20, 1988, and full text by Apr. 30, 1989, to Jerzy Swiatek, Technical Univ. of Wrocław, Inst. of Control and Systems Engineering, Janiszewski St. 11/17, 50-370 Wrocław, Poland.

IEEE Pacific Rim Conf. on Communications, Computers, and Signal Processing: June 1-2, 1989, Victoria, B.C., Canada. Sponsors: IEEE Victoria Section, Univ. of Victoria. Submit manuscript by Dec. 30, 1988, Technical Program Committee, c/o Warren D. Little, Dept. of Electrical and Computer Engineering, Univ. of Victoria, PO Box 1700, Victoria, B.C., Canada V8W 2Y2, phone (604) 721-8686.

IEEE Computer Graphics and Applications seeks papers for a special issue in early 1990 on graphics for electronic CAD, CAM, CAE, and CIM. Submit tentative abstract by Jan. 1, 1989, and manuscript by Mar. 1, 1989, to Ronald Backer or Martin Snelgrove, Computer Systems Research Inst., Univ. of Toronto, 10 Kings College Rd., Rm. 2002, Toronto, Ont., M5S 1A4, Canada.

Third AI Research in Environmental Science Workshop: May 2-4, 1989, Washington, DC. Submit abstract by Jan. 1, 1989, to William R. Moninger, Environmental Research Labs, NOAA, R/E2, 325 Broadway, Boulder, CO 80803, phone (303) 497-6435.

Third Parallel Processing Symp.: Mar. 29-31, 1989, Fullerton, Calif. Sponsors: Orange County IEEE Computer Society Chapter, California State Univ., Fullerton. Submit abstract by Jan. 6, 1989, and paper by Feb. 6, 1989, to Larry Carter, 1619 N. Hale, Fullerton, CA 92631, phone (714) 738-3414.

Working Conf. on Engineering for Human-Computer Interaction: Aug. 21-25, 1989, Napa Valley, Calif. Sponsor: IFIP. Submit paper by Jan. 9, 1989, to Leonard Bass, Software Engineering Inst., Carnegie Mellon Univ., Pittsburgh, PA 15213; or Claus Unger, Praktische Informatik II, Univ. of Hagen, Feithstr 140, D-5800 Hagen, West Germany.

Second Int'l Conf. on Solid State and Integrated Circuit Technology: Oct. 22-28, 1989, Beijing. Sponsors: Univ. of California Extension, Berkeley, and the Chinese Inst. of Electronics. Submit abstract by Jan. 9, 1989, to Linda Reid, Continuing Education in Engineering, Univ. of California Extension, 2223 Fulton St., Berkeley, CA 94720, phone (415) 642-4151.

Working Conf. on Computer Aided Design Systems Using Artificial Intelligence Techniques: June 6-7, 1989, Tokyo. Cosponsors: IFIP, IPSJ. Submit paper by Jan. 10, 1989, to Gotaro Odawara, c/o Business Center for Academic Societies Japan, 3-23-1 Hongo, Bunkyo-ku, Tokyo 113, Japan.

ICAAIL 89, Second Int'l Conf. on Artificial Intelligence and Law: June 13-16, 1989, Vancouver, Canada. Submit extended abstract

by Jan. 10, 1989, and final paper by Apr. 15, 1989, to Edwina L. Rissland, Computer and Information Science Dept., Univ. of Massachusetts, Amherst, MA 01003, phone (413) 545-0332.

First IEEE Symp. on Parallel and Distributed Processing: May 22-23, 1989, Dallas. Sponsor: Dallas Section of the IEEE Computer Society. Submit paper by Jan. 15, 1989, to Mark Shadows, Information Technologies Lab, Texas Instruments, PO Box 655474, MS 238, Dallas, TX 75265, or call Behrooz Shirazi at (214) 692-2874.

IFCS 89, Second Conf. of the Int'l Federation of Classification Societies: June 27-30, 1989, Charlottesville, Va. Sponsor: IFCS. Submit abstract by Jan. 15, 1989, to Robert F. Ling, Mathematical Sciences Dept., Clemson Univ., Clemson, SC 29634-1907.

ICIP 89, Int'l Conf. on Image Processing: Sept. 5-8, 1989, Singapore. Sponsors: IEEE Singapore Section, National Univ. of Singapore. Submit extended summary by Jan. 16, 1989, to Ngan King-Ngi, c/o Meeting Planners Pte. Ltd., 100 Beach Rd., No. 33-01 Shaw Towers, Singapore 0718, Republic of Singapore, phone (65) 297-2822.

VLSI 89, Int'l Conf. on Very Large Scale Integration: Aug. 16-18, 1989, Munich, West Germany. Sponsor: IFIP. Submit paper by Jan. 16, 1989, to Carlo H. Sequin, Electrical Engineering and Computer Sciences Dept., Computer Science Div., Univ. of California, Berkeley, CA 94720 (for the Americas); Mike Newman, Commission of the European Communities, Rue de la Loi 200, A25, 6/12, B-1049, Brussels, Belgium (for Europe and Africa); or Takayuki Yanagawa, NEC, 1753 Shimonumabe, Nakahara-ku, Kawasaki 211, Japan (for Asia and Australasia).

EKAW 89, Third European Knowledge Acquisition Workshop for Knowledge-Based Systems: July 3-7, 1989, Paris. Submit extended abstract by Jan. 25, 1989, to Jean G. Ganascia, LAFORIA, Univ. Pierre et Marie Curie, Tour 45-46, 4 Place Jussieu, 75230 Paris Cedex 05, France, phone 32 (1) 69-41-66-26.

20th Pittsburgh Conf. on Modeling and Simulation: May 4-5, 1989, Pittsburgh, Pa. Sponsor: Univ. of Pittsburgh. Submit summary and abstract by Jan. 31, 1989, to William G. Vogt or Marlin H. Mickle, 348 Benedum Engineering Hall, Univ. of Pittsburgh, Pittsburgh, PA 15261.

Sixth IEEE Workshop on Real-Time Operating Systems and Software: May 11-12, 1989, Pittsburgh. Cosponsors: Carnegie Mellon Univ. Submit position paper by Feb. 1, 1989, to John B. Goodenough, Software Engineering Inst., Carnegie Mellon Univ., Pittsburgh, PA 15213, phone (412) 268-6391.

IEEE Int'l Conf. on Computer Design: Oct. 2-4, 1989, Cambridge, Mass. Submit summary by Feb. 1, 1989, to Sumit Das

Gupta, IBM, Bldg. 306, ZIP 3A1, Hopewell Junction, NY 12533, phone (914) 894-0540.

Compsec 89: Sept. 18-22, 1989, Kissimmee, Fla. Submit paper and panel proposal by Feb. 1, 1989, Stanley Y.W. Su, 301 Computer Science and Engineering Bldg., Database Systems Research and Development Center, Univ. of Florida, Gainesville, FL 32611, phone (904) 335-8458 or 8460.

Network Management and Control Workshop: Sept. 19-21, 1989, Tarrytown, NY. Sponsors: IEEE, et al. Submit paper by Feb. 1, 1989, to Basil Maglaris, Center for Advanced Technology in Telecommunications, Polytechnic Univ., 333 Jay St., Brooklyn, NY 11201, phone (718) 260-3050.

CSM 89, Conf. on Software Maintenance: Sept. 25-28, 1989, Pensacola, Fla. Submit paper and panel session proposal by Feb. 15, 1989, to Capt. Thomas Pigoski, NSGID Pensacola, Corey Station, Pensacola, FL 32511, phone (804) 452-6399.

IEEE Software: November 1989. Articles are sought for a special edition on reverse engineering. Submit manuscript by Mar. 1, 1989, to Elliott Chikofsky, Index Technology Corp., 1 Main St., Cambridge, MA 02142, phone (617) 494-8200.

IEEE Trans. on Computers: December 1989. A special issue is planned on computer systems performance. Submit manuscript by Mar. 1, 1989, to Edward D. Lazowska, Computer Science Dept. FR-35, Univ. of Washington, Seattle, WA 98195, phone (206) 543-4755.

Int'l Journal of Approximate Reasoning plans a special issue on belief functions and belief maintenance in artificial intelligence. Submit paper by Mar. 1, 1989, to Prakash P. Shenoy, School of Business, Univ. of Kansas, Summerfield Hall, Lawrence, KS 66045-2003, phone (913) 864-7551; or Gautam Biswas, Dept. of Computer Science, Box 1688, Station B, Vanderbilt Univ., Nashville, TN 37235, phone (615) 343-6204.

IEEE Software: January 1990. A special issue is planned on software development metrics. Submit paper by Apr. 15, 1989, to Peter Dyson, Software Productivity Solutions, 122 N. Fourth Ave., Indianapolis, FL 32903, phone (407) 984-3370.

Second IEEE Workshop on Workstation Operating Systems: Sept. 27-28, 1989, Pacific Grove, Calif. Submit position statement by Apr. 15, 1989, to Luis-Felipe Cabrera, Mail Code K52/803, IBM Almaden Research Center, 650 Harry Rd., San Jose, CA 95120-6099.

Fourth Int'l Workshop on High-Level Synthesis: Oct. 15-18, 1989, Kennebunkport, Maine. Cosponsor: ACM. Submit extended abstract by June 16, 1989, to Raul Camposano, IBM Research Div., T.J. Watson Research Center, PO Box 218, Yorktown Heights, NY 10598, phone (914) 945-3971.



IEEE COMPUTER SOCIETY

IEEE COMPUTER SOCIETY

Membership / Subscription Application



THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

BENEFITS



Computer

Computer comes automatically with membership. Written, reviewed, and refereed by experts, it features survey and tutorial articles covering the entire computer field, and departments such as new products, new product reviews, standards, and a reader forum called "The Open Channel!" (monthly).

Technical Committees

Participate in one of more of our 33 technical committees — networks of professionals with common interests in specialty areas within computer hardware, software, and applications.

Standards Working Groups

Participate in the development of the more than 100 standards projects currently sponsored by the society in such diverse areas as software engineering, local area networks, microprocessor buses, design automation, programming languages, and standards definitions.

Computer Society Press Books

Receive discounts of up to 50% on over 600 titles covering a broad spectrum of computer science topics such as networking, communications, advanced systems, image processing, security, artificial intelligence, and design automation. Over 60 new titles are published annually.

Conferences and Tutorials

Choose from more than 100 conferences annually, ranging from large industry-oriented conferences replete with exhibits to small, highly interactive workshops. Members receive special low rates.

Schedule of Fees

Membership dues and periodical subscriptions are annualized to, and expire on, December 31. Choose full or half-year rate schedules depending on date of receipt by the Computer Society as indicated below.

To join: see item 1, 2, or 3.

To subscribe: see item 4.

1 I don't belong to the IEEE and I want to join just the Computer Society ☐ \$19.50 ☐ \$39.00

2 I don't belong to the IEEE and I want to join both the Computer Society and the IEEE*
(Total amount to be paid includes annual dues, and regional assessment, if any.)

I reside in Region 1 (6 (United States))..... ☐ \$44.00 ☐ \$88.00
I reside in Region 7 (Canada)..... ☐ \$41.00 ☐ \$82.00
I reside in Region 8 (Europe, Africa, or the Middle East)..... ☐ \$41.50 ☐ \$83.00
I reside in Region 9 (Latin America)..... ☐ \$36.00 ☐ \$72.00
I reside in Region 10 (Asia and Pacific)..... ☐ \$37.00 ☐ \$74.00

*ACM members who join both IEEE and the Computer Society may deduct \$5 off the full-year rate; \$2.50 off the half-year rate.

3 I already belong to the IEEE and I want to join the Computer Society. ☐ \$ 7.50 ☐ \$15.00
IEEE Member Number _____

4 OPTIONAL PERIODICALS for new or current members

	ISSUES per year	☐ \$ 9.50	☐ \$19.00
IEEE Computer Graphics and Applications (3061).....	6	☐ \$10.00	☐ \$20.00
IEEE Design and Test (3111).....	6	☐ \$ 6.00	☐ \$12.00
IEEE Expert (3151).....	4	☐ \$ 9.00	☐ \$18.00
IEEE Micro (3071).....	6	☐ \$ 9.00	☐ \$18.00
IEEE Software (3121).....	6	☐ \$10.00	☐ \$20.00
Transactions on Computers (1161).....	12		
Transactions on Knowledge and Data Engineering (1471).....	4	☐ \$ 5.00	☐ \$10.00
Transactions on Pattern Analysis and Machine Intelligence (1351).....	12	☐ \$10.00	☐ \$20.00
Transactions on Software Engineering (1171).....	12	☐ \$10.00	☐ \$20.00

Total amount remitted with this application \$ _____

☐ Checks are accepted in Belgian, British, German, Swiss, Japanese, or U.S. currencies. U.S. checks must be drawn on a U.S. bank.

☐ Visa ☐ MasterCard ☐ American Express ☐ Eurocard ☐ Diners Club

--	--	--	--	--	--	--	--	--	--

Charge Card Number

Mo.	Yr.

Exp. Date

I hereby make application for Computer Society membership and, if elected, will be governed by IEEE's and the society's constitutions, bylaws, and statements of policies and procedures.

MAILING ADDRESS

Full signature _____

Date _____

First name _____ Middle Initial(s) _____ Last name _____

Street address _____ City _____ State/Country _____ Zip _____

EDUCATION (highest level completed)

Course _____ Degrees received _____ Date _____

Name of educational institution _____ Date of birth _____ ☐ Male ☐ Female

OCCUPATION

Title or Position _____

REFERENCE (an IEEE member; if unknown, a managerial person who knows you professionally)

Firm name _____ Name (print in full) _____ IEEE Member No. (if applicable) _____

Firm address _____ Street address _____

City _____ State/Country _____ Zip _____ City _____ State/Country _____ Zip _____

Return with your remittance to:

Residents of Europe mail to:

Asia/Pacific residents mail to:

IEEE Computer Society, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-2578 USA

IEEE Computer Society, 13, Avenue de l'Aiglon, B-1200, Brussels, BELGIUM

IEEE Computer Society, Ooshima Building, 2-19-1 Minami-Aoyama, Minato-ku, Tokyo 107 JAPAN

PC-1188

CAREER OPPORTUNITIES

RATES: \$12.00 per line, \$120 minimum charge (up to ten lines). Average five typeset words per line, eight lines per column inch. Add \$10 for box number. Send copy at least one month prior to publication to Heidi Rex or Marian Tibayan, Classified Advertising, *COMPUTER Magazine*, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720; (714) 821-8380.

In order to conform to the Age Discrimination in Employment Act and to discourage age discrimination, *COMPUTER* may reject any advertisement containing any of these phrases or similar ones: "...recent college grads...", "...1-4 years maximum experience...", "...up to 5 years experience...", "...10 years maximum experience..." *COMPUTER* reserves the right to append to any advertisement, without specific notice to the advertiser, "Experience ranges are suggested minimum requirements, not maximums." *COMPUTER* assumes that, since advertisers have been notified of this policy in advance, they agree that any experience requirements, whether stated as ranges or otherwise, will be construed by the reader as minimum requirements only.

PENNSYLVANIA STATE UNIVERSITY Director, Computer Engineering Program

Penn State—The Electrical Engineering Department at The Pennsylvania State University is inviting applications for the position of Director of its Computer Engineering Program. Applicants should have an established reputation in Computer Engineering, strong leadership qualities, and ability to attract funding for faculty research.

Approximately 12 of the 60 Department faculty are members of the program. Recent years have seen the introduction of separate Bachelor's, Master's, and Ph.D. degrees in Computer Engineering. Enrollment in these programs is expected to grow substantially, particularly at the graduate level. Research thrusts exist currently in Parallel and Distributed Processing, Interconnection Networks, Fault Tolerant Computing, Image Processing, Database Systems, and Computer Communication. The new Director should strive to focus these research activities into areas where the program can establish national prominence.

Nominations will be accepted until January 31, 1988, or until suitable qualified candidates are selected. Inquiries should be directed to the Computer Engineering Director Search Committee, Room 129, Electrical Engineering East Building, Box-CS, The Pennsylvania State University, University Park, PA 16802.

An Affirmative Action/Equal Opportunity Employer. Women and Minorities Encouraged to Apply.

THE GEORGE WASHINGTON UNIVERSITY Computer Science Faculty Positions

Computer Science faculty positions at the Assistant, Associate and Full Professor ranks are available commencing Fall Semester 1989.

Especially sought are applicants able to conduct research and teach in the overlapping areas of image processing, machine vision and artificial intelligence. Candidates should have an earned Doctorate and research experience with an interest in both teaching and research. Ability to attract funded research is valued. The George Washington University is located in the center of Washington, DC. This metropolitan area sustains the second largest concentration of research and development activity in the United States, creating a continuing demand for rigorously-trained engineers and many research opportunities. Our Department is the largest in the School of Engineering & Applied Science with 38 full-time faculty, accredited electrical engineering, computer engineering, and computer science degree programs, a large graduate and undergraduate student body, and a substantial research budget. Current funded research projects include computer graphics, computer security, data transmission standards and compression, fast packet switching, image processing, intelligent user interfaces, laser shields, MHD plants, magnetic devices, medical imaging, multipath fading and encryption, remote sensing, space-based radar, and special-purpose VLSI designs. Send curriculum vita, list of publications and 3 references to:

Professor James D. Foley, Chairman
Department of Electrical Engineering &
Computer Science
School of Engineering & Applied Science
THE GEORGE WASHINGTON
UNIVERSITY
Washington, DC 20052

The George Washington University is an equal opportunity/affirmative action employer.

UNIVERSITY OF CALIFORNIA AT DAVIS Faculty Positions in Computer Science

The Computer Science Division of the Department of Electrical Engineering and Computer Science of the University of California at Davis invites applications for several tenure-track positions at all ranks. The primary areas of interest are computer architecture, networks, VLSI design, and parallel computing. Outstanding candidates in other fields will also be considered.

The Division of Computer Science is in a period of rapid growth, with the aim of becoming one of the leading computer science programs in the nation. The division offers programs leading to Bachelor of Science,

Master of Science, and Doctor of Philosophy degrees. Its expanding research facilities include an Encore Multimax processor, several VAX systems (8600, 785, etc.), two Isis graphics workstations, a Ridge 32, numerous Sun, Microvax II, and Apollo workstations, and a number of special purpose systems all linked through Ethernet and serial communications networks. Other LAN systems are available for instruction and research. General purpose campus facilities are available, and the supercomputer resources of the Lawrence Livermore National Laboratories are also accessible. Our College of Engineering is the nation's sixteenth largest producer of Ph.D.'s in a University which has the nineteenth largest extramural research funding. Salary and benefits are extremely attractive.

The town of Davis has a population of approximately 50,000. Located eleven miles from Sacramento, it is situated within driving distance from the major cultural centers of the San Francisco area as well as the outstanding recreational facilities of the Sierra Nevada. The town has earned an international reputation for its progressive energy-conservation policies. Because of its location, its climate, and its small town character, Davis is considered by many to be a fine living environment.

We are seeking individuals with strong records of teaching and research with ambitious plans. Senior appointments require outstanding records of achievement; junior appointments must show evidence of great promise. All faculty are expected to have a strong commitment to teaching at all degree levels, and to demonstrate the ability to attract significant research support. The positions require a Ph.D. degree or equivalent, and are open until filled. Send a resume and the names of at least three references to:

Richard F. Walters, Chair
Division of Computer Science

or

S. Louis Hakimi, Chair
Department of Electrical Engineering
and Computer Science
University of California
Davis, CA 95616

The University of California is an equal opportunity/affirmative action employer.

COLUMBIA PACIFIC UNIVERSITY

FULLY APPROVED UNIVERSITY DEGREES! Economical home study for Bachelor's, Master's, Ph.D., fully approved by California State Department of Education, not accredited. Prestigious faculty counsels for independent study and life experience credits (5000 enrolled students, 400 faculty). **Free information**—Richard Crews, M.D. (Harvard), President, Columbia Pacific University, Department 3F6N, 1415 Third Street, San Rafael, CA 94901. Toll free: (800) 227-0119; California: (800) 552-5522; or (415) 459-1650.

UNIVERSITY OF CALIFORNIA, DAVIS
Chair, Division of Computer Science

Nominations and applications are invited for the position of Chair, Division of Computer Science at the University of California, Davis. The chair must be a distinguished researcher with a prominent research record and an accomplished teacher. The chair is expected to provide both intellectual and administrative leadership.

The Division of Computer Science is a semi-autonomous unit within the Department of Electrical Engineering and Computer Science. The Division of Computer Science has 17 faculty members and almost 100 full time graduate students. The entire department has a total of 46 faculty members and almost 250 full time graduate students. The Division of Computer Science has experienced rapid growth over the past five years and significant further growth is expected over the next five years. The new chair is expected to have at least 7.9 new faculty positions in the next five years. Substantial startup funds are available for equipment for the chair and for new recruits. As part of the growth plans significant amounts of equipment and space have been committed to the Division of Computer Science. A new Engineering building including approximately 110K sq. ft. is scheduled for the EECSS Department. In addition, a new 32K sq. ft. Engineering Research building, donated by regional industry, will be available for the research activities of the department.

Davis is a very pleasant, family-oriented community. The 900 acre, tree shaded central campus has about 20,000 students. The City of Davis enjoys the mild climate of the Sacramento Valley, and is within easy driving distance of the Sierra Nevada mountains, San Francisco, Napa Valley, and Silicon Valley. In addition, the department provides salary and benefits that are extremely attractive.

The Chair position is available July 1, 1989. Screening of nominations and applications begins October 15, 1988, and will continue until the position is filled. Please send a resume listing the names of at least three references to:

Professor Charles Martel, Chair
Search Committee

Division of Computer Science
University of California
Davis, CA 95616

Phone: (916) 752-2651

CSNET: martel@ucd.csnet@relay.cs.net
ARPANET: martel@ucdavis.edu.

The University of California, Davis, is an equal opportunity/affirmative action employer.

LOUISIANA STATE UNIVERSITY
IN SHREVEPORT

Two Senior Positions Available. (1) Director, Master's program in Systems Technology. (2) Computer Science Faculty position. Assistant Professor or above, for teaching/research and/or Department Chairmanship. Call 1-318-797-5311 for complete information. C.A. Hall, Acting Chairman, CSC Dept., LSU-S, One University Place, Shreveport, LA 71115.

An Equal Opportunity/Affirmative Action Employer.

GEORGE MASON UNIVERSITY
Faculty Positions in
Computer Science

George Mason University is located in Fairfax County, Virginia near the outstanding scientific, cultural, and touristic attractions of Washington, D.C. The university has made a major commitment to establish a School of Information Technology and Engineering (SITE) and maintain an outstanding academic and research program in the Department of Computer Science, as well as related research centers in software engineering, artificial intelligence, parallel computation, and computer vision.

Tenure track and visiting faculty positions are available at all ranks, with the nominal date of appointments commencing September 1989. Candidates for these positions must have an earned doctorate in a relevant field, along with professional accomplishments that are appropriate to the rank sought. They should be interested in both teaching and developing a strong sponsored research program.

The relevant research and teaching areas of interest in Computer Science are: computer communications and networks, software engineering, parallel computation, computer vision, artificial intelligence, and distributed systems.

There are over 80 full-time faculty in SITE, and over 20 full-time faculty in Computer Science. There are numerous opportunities for government and industrial interaction in this suburban high technology area, just 15 miles southwest of the Nation's Capital.

For full consideration, please send a detailed resume together with the names of four references, and other supporting material to: Professor Harry Wechsler, Chair, Recruitment Committee, Department of Computer Science, George Mason University, Fairfax, VA 22030, or call at 703-323-2713. Applications are sought by Feb. 1, 1989, and these will receive first priority. AA/EOE.

UNIVERSITY OF TEXAS
AT ARLINGTON

The College of Engineering seeks applications and nominations for the Moncrief O'Donnell Chair in Artificial Intelligence for Manufacturing. The distinguished individual chosen to fill this endowed chair will lead research and academic programs and serve as an Associate Director for the Automation and Robotics Research Institute (ARRI).

ARRI is a new facility dedicated in February 1988 as a result of a \$10 million gift from the Fort Worth and Dallas communities. The 48,000 sq. ft. facility is located a short distance from the UTA campus. The campus is at the center of the DFW Metroplex, an international aerospace and manufacturing hub. The selectee will be appointed a full Professor with tenure in one of the academic departments or programs of the College (with aerospace, computer science, electrical, industrial, or mechanical engineering

being most likely). Joint appointments are feasible, if appropriate.

Duties will include providing scientific and technical leadership in the Institute, collaborating with industry in applied research and development efforts at the Institute, overseeing the Artificial Intelligence for Manufacturing Laboratory and its professional staff, developing and conducting a program of advanced research, supervising graduate students, teaching graduate courses, mentoring junior faculty, and other responsibilities normally assumed by professors of senior rank.

Qualifications include an earned PhD in an appropriate discipline; international recognized credentials in the applications of artificial intelligence to manufacturing; exceptional leadership skills; personal qualities and interest to work in an interdisciplinary team environment; proven capability to develop and manage extramurally funded research projects; and achievements and publications warranting appointment as a full Professor. Teaching experience and research experience in industry and/or government are desirable. A competitive salary will be provided.

Applicants should submit a detailed resume addressing the above requisites and the names of at least five references to: Dr. John H. McElroy, Dean of Engineering and Chairman of Search Committee, Box 19019, The University of Texas at Arlington, Arlington, TX 76019. The immigration status of a non-citizen must be stated in the resume. Applications received by January 2, 1989 will receive full consideration. The University of Texas at Arlington is an Equal Opportunity/Affirmative Action Employer.

UNIVERSITY OF CALIFORNIA
SAN DIEGO

The Department of Computer Science and Engineering expects to have junior level faculty positions (tenure track) in computer science and computer engineering. These positions involve research and teaching at both graduate and undergraduate levels. The department and the university offer an outstanding environment for academic computer science. We are interested in strong candidates from all major areas of computer science (except numerical analysis). For appointment at the assistant professor level, evidence of excellent potential for conducting research in computer science is necessary. Salary and rank will be commensurate with qualifications in conformance with University of California policies; a Ph.D. (or advancement to candidacy) in computer science or computer engineering is required. Applications received by January 31, 1989 will be considered for all positions, but later applications will also be considered if unfilled positions remain. Please send a curriculum vitae and the names of four references to:

Dr. Christos H. Papadimitriou, Chair
Faculty Recruitment Committee
Dept. of Computer Science and

Engineering

Mail Code C-014

University of California, San Diego
La Jolla, California, 92093

UCSD is an Equal Opportunity/Affirmative Action employer.

**VIRGINIA POLYTECHNIC
INSTITUTE & STATE UNIVERSITY**
Bradley Department of
Electrical Engineering

Applications are invited for tenure track positions at all levels in the Computer Engineering degree program of the Bradley Department of Electrical Engineering at Virginia Tech. Individuals with relevant software and/or hardware backgrounds are encouraged to apply. The ability to teach undergraduate and graduate courses effectively in computer architecture, logic design, and microprocessor system design is desired. Other course areas include local area and long haul network design, fault tolerance and testing, VLSI design, computer aided design, pattern recognition, and computer vision. Candidates are expected to develop an externally funded research program. Current research projects are in the areas of VLSI design and test, fault tolerant computer architectures, chip level modeling, microprocessor system architectures, pattern recognition, and computer vision. Virginia's Center for Innovative Technology provides funds for cooperative research projects between high technology companies and Virginia universities. Departmental computing facilities include a VAX 11/785, VAX 11/750, 10 Vaxstations, Harris H-800, MV10000, HP9000, HP350, HP550 and workstation network. LAN connections provide access to other VAX-8000 processors and an IBM 3090. All Virginia Tech engineering students are required to purchase an IBM PC. Each faculty member is provided a PC for his/her use. The Electrical Engineering Department is housed in a newly completed six story building which provides extensive, well designed space for teaching and research laboratories. Applicants should possess an earned doctorate. Virginia Tech is Virginia's land grant university with approximately 23,000 students. It is located in the Appalachian Mountains with many outdoor activities available. Send resume, including names, addresses, and phone numbers of three references and employment/citizenship status to: Dr. Warren Stutzman, Personnel Committee Chairman, Bradley Department of Electrical Engineering, Virginia Tech, Blacksburg, VA 24061. Applications will be accepted until April 1, 1989, or until suitable candidates are selected. Virginia Tech is an Affirmative/Action Equal Opportunity Employer.

PRINCETON UNIVERSITY

The Department of Electrical Engineering invites applications for a full time, tenure-track faculty position. The discipline of particular interest is Computer Engineering with a specialization in an area such as: CAD for integrated circuits and systems, fault tolerance and computer architecture, VLSI, automated synthesis of digital systems. Please send a complete resume, a description of research and teaching interests and names of three references to Professor Stuart Schwartz, Chairman, Dept. of EE, Princeton University, Princeton, NJ 08544. An Affirmative Action/Equal Opportunity Employer.

ACADEMIA SINICA
Taiwan, Republic of China
Institute of Information Science

Applications are invited for research positions in Institute of Information Science, Academia Sinica. Ph.D. in computer related fields required. Demonstrable research ability necessary. Applicants for senior positions must have proven research record. All fields in computer science are welcome.

The Institute offers a good research environment. No duty of teaching. Facilities include 3 VAX's, 4 SUN's (including a SUN-4/260), 2 E&S workstations, a lot of small computers and an easily accessible ETA-10Q super computer at Academia Sinica. Two-year expansion plan will strengthen our facilities with an IRIS and a multi-processor machine.

Interested people please send application to Dr. Y. S. Kuo, Acting Director, Institute of Information Science, Academia Sinica, Taipei, Taiwan, Republic of China. e-mail address: WT6B0001@TWNMOE10.BIT-NET. FAX: (011-886-2) 782-4814.

**CALIFORNIA STATE UNIVERSITY,
HAYWARD**
Department of Mathematics and
Computer Science

The department is now seeking applicants for a tenure track Assistant Professor position in Computer Science beginning Fall, 1989.

Applicants should have the Ph.D. in Computer Science, or in a related field with computer science experience, and should have a commitment to excellence in teaching, willingness and ability to participate in curriculum development, and competence and potential for significant professional activities, including research and publication. All areas of specialization will be considered, but the areas of architecture, networks, software system design are particularly sought. The interests of the present faculty include a wide range of theory, software design, and hardware design topics.

The department has a Pyramid 9805 mini-computer and four Sun 3/160 color workstations connected on an Ethernet, a graphics lab, an interactive classroom, a digital and a microprocessor lab, and numerous IBM PC's for faculty use. In addition, the campus has a Cyber 170/720, a Sequent Symmetry 27, a PRIME 9755, and several PC labs for student use.

Interested applicants should send a resume and three references to:

CS Faculty Search Committee
Department of Mathematics and
Computer Science
California State University, Hayward
Hayward, CA 94542-3092
e-mail: csuhfaculty-search@111-crg.arsa

Applications received by January 1, 1989, will be assured full consideration. Applications will be accepted as long as positions remain available.

California State University, Hayward, an Equal Opportunity, Affirmative Action employer, seeks to create a stimulating atmosphere for its ethnically diverse student body and encourages applications from women and men of all ethnic backgrounds and physical abilities.

THE OHIO STATE UNIVERSITY
Department of Computer
and Information Science

Applications are invited for faculty positions at all levels. A Ph.D. in computer science or a closely related field is desired. Candidates in the areas of artificial intelligence, computer graphics, databases, programming languages operating systems, parallel and distributed computing, software engineering, and VLSI design will be considered. Of special interest are senior researchers in the areas of computer architecture, computer graphics, and software systems.

The Laboratory for AI Research in the Department has been awarded a State of Ohio Regent's Academic Excellence grant for the cognitive science-related aspects of Artificial Intelligence. Candidates in computational theories of vision and speech, theoretical studies on architectures for intelligence and conceptual information processing are invited to apply under this program.

Computing facilities within the Department include 200 networked Sun-3 workstations, 8 Hewlett Packard color graphics workstations, and over 300 Macintoshes. All CIS majors use workstations in their course work, and many are also used for department research. Other Departmental computing facilities include a DEC-2060, five Pyramids, 10 IBM PCs, 15 Xerox LISP machines, an Intel iPS/5 Hypercube, an Encore Multimax, a BBN Butterfly, and numerous systems for graphics, robotics, etc. The OSU Computing Center facilities include a Cray XMP, IBM 3081-D, and several IBM 4341s. All systems are attached to a campus-wide fiber optic network (Pronet 80) using TCP/IP protocols. Access is available to the national networks (ARPANET, NSFNET, BITNET, USENET).

To apply, please send application and resume to Prof. Ming T. (Mike) Liu, Chairman of Faculty Search Committee, Department of Computer and Information Science, The Ohio State University, 2036 Neil Avenue, Columbus, Ohio 43210-1277. (NSFNET Address: Liu@OSU-20.IRCC.Ohio-State.Edu). In order to expedite consideration of your application, please ask three references to send their confidential letters directly to Prof. Liu. The Ohio State University is an equal opportunity/affirmative action employer.

TRINITY COLLEGE

The Department of Engineering and Computer Science invites applications from outstanding candidates for a tenure-track position at the Assistant Professor-level commencing September, 1989, in the areas of FLUID MECHANICS/HEAT TRANSFER or ROBOTICS/CONTROLS. Experimental background highly desirable. The position involves graduate and undergraduate instruction and research, and a doctoral degree is a prerequisite. We are interested in receiving applications from qualified women and minorities. Trinity College is an equal opportunity/affirmative action employer. Please send resume to Professor Joseph D. Bronzino, Chairman, ECS Department, Trinity College, Hartford, CT 06106. Applications will be accepted until February 15, 1989.

MISSISSIPPI STATE UNIVERSITY
Head, Department of
Computer Science

Mississippi State University invites applications and nominations for the position of Head of the Computer Science Department. Mississippi State is seeking an individual who can provide leadership and direction for its computer science program offering B.S., M.S., and Ph.D. degrees. The successful candidate must have a Ph.D. degree in computer science or a closely related discipline and must have faculty experience in a doctoral granting computer science department and an established record in research.

Mississippi State University is a comprehensive land grant university with teaching, research and service programs spanning a broad range of disciplines and professions. The Computer Science Department is a part of the College of Arts and Sciences and has fifteen full-time faculty. The new department head will be expected to recruit and fill current open positions. The Research Center for Advanced Scientific Computing has recently received \$12 million grant and the University recently created a Center for Robotics, Automation and Artificial Intelligence. A comprehensive research program is now a principal priority of the University with new programs being established and underway.

A central computer facility consisting of a four-processor Unisys 1100 along with various peripheral devices and computers is available to the students and faculty. The Computer Science Department maintains several computer laboratories consisting of microcomputers, minicomputers, SUN workstations, and access to a VAX 11/755. The Computer Science Department will be housed in a separate new facility now undergoing a \$2 million renovation, to be available in 1989.

Mississippi State is located in Starkville, Mississippi, a college town of approximately 20,000. Memphis, Tennessee, Birmingham, Alabama, and Jackson, Mississippi, are within two to three hours driving time. A regional airport serves Starkville and nearby cities of Columbus and West Point.

Interested applicants please forward vita and names of four references to:

Charles L. Bradshaw
Acting Department Head
Computer Science Department
Mississippi State, MS 39762
601-325-2756

**MISSISSIPPI STATE UNIVERSITY IS
AN EQUAL OPPORTUNITY/AFFIRMATIVE
ACTION EMPLOYER.**

DARTMOUTH COLLEGE
Thayer School of Engineering
Image Processing

Thayer School of Engineering at Dartmouth College invites nominations and applications at the assistant or associate professor level in Image Processing. Applicants should have a doctorate in engineering or applied mathematics, or closely-related field. This faculty member will be involved in undergraduate and graduate teaching and is expected to establish an ongoing funded

research program. There are several faculty members and active projects in biomedical image processing, as well as opportunities for research collaboration with the Dartmouth-Hitchcock Medical Center, the Departments of Earth Sciences and Mathematics and Computer Science, and the U.S. Army Cold Regions Research and Engineering Laboratory.

Campus computing facilities include a distributed system of networked UNIX workstations; local supercomputing on Convex and Alliant machines; and an Internet gateway providing access to NSF and other supercomputing centers worldwide.

Thayer School is an interdisciplinary school of engineering, offering all degrees through the doctorate, and is in the midst of a major expansion in facilities, faculty, students, and research support. Dartmouth College is a highly-selective university composed of the undergraduate liberal arts college and graduate and professional programs through the doctoral level in arts and sciences, medicine, business, and engineering.

Dartmouth is located in a small New England town at the junction of the Connecticut River and the Appalachian Trail, in an area noted for excellent outdoor recreational opportunities. Cultural activities revolve around the Hopkins Center for the Performing Arts, the Hood Museum of Art, and numerous local music and theater groups. Hanover is served by a local airport with service to Boston and New York, and by interstate highways, rail and bus lines.

Interested applicants should submit a resume and names/addresses of 3 references to:

Dr. Carol B. Muller, Assistant Dean
Thayer School of Engineering
Dartmouth College, Hanover, NH 03755
Review of applications will begin November 15, 1988.

Dartmouth College is an equal opportunity/Affirmative Action employer and encourages applications from women and members of minority groups.

CLEVELAND STATE UNIVERSITY
Computer
Instructor or Asst. Professor

The Department of Computer and Information Science at Cleveland State University has a tenure track position in the areas of distributed systems or software engineering and design. Responsibilities include teaching (two courses/quarter) & research. Salary is very competitive. Qualifications: Ph.D. in Computer Science, MIS, or a closely related field. Ph.D. candidates with substantial progress on a dissertation will be considered. Close relationships to business, engineering, and other departments provide an environment conducive to research and consulting. In addition to university computing facilities, the department has laboratories using VAX computers running UNIX, VMS, and state of the art software. Faculty offices are equipped with personal computers. Inquiries and vita should be sent to: Dr. Thomas S. Heines, Chairperson, Computer & Information Science Dept., Cleveland State University, E. 24th & Euclid Ave., Cleveland, OH 44115. Equal Opportunity Employer, m/f/h.

UNIVERSITY OF MARYLAND
Chairperson and Professor
Computer Science Department

The University of Maryland at College Park invites applications and nominations for the position of Chairperson and Professor of its Computer Science Department. Candidates for the position should have a well-established reputation in research and teaching in computer science, and have demonstrated ability to lead a major research department. The Department has forty-five professorial positions and offers undergraduate and graduate (M.S. and Ph.D.) degrees. Many faculty hold joint research appointments in the University's Institute for Advanced Computer Studies. The prospective appointment date is approximately July 1, 1989. Applications received by January 15, 1989 will receive full consideration. Inquiries, nominations, and applications should be addressed to: Chairman, Search Committee, Office of the Dean, College of CMPS, Room 2300 Mathematics, University of Maryland, College Park, Maryland 20742.

The University of Maryland is an Equal Opportunity, Affirmative Action Employer.

GEORGIA STATE UNIVERSITY
Computer Information Systems
Faculty Positions

The Department of Computer Information Systems invites applications for tenure-track faculty positions at the rank of Instructor, Assistant, Associate and Professor in Computer Information Systems. The department currently seeks faculty with strong research and teaching capabilities in the following areas: Software engineering, programming languages, computer architecture, operating systems, data structures, and office information systems, but welcomes applications from other areas as well.

Candidates at the associate/professor level must have a Ph.D. or equivalent. Candidates at the assistant level must have completed all requirements for the doctorate by the time their appointment becomes effective. Candidates at the instructor level must have completed their master's program.

The department is part of the College of Business Administration, has BBA and Master's programs as well as a doctoral program with approximately 29 Ph.D. candidates and includes 20 full-time faculty members. It is located in the heart of the Atlanta business and financial community.

Computing resources include IBM, AMDahl, and Unisys mainframes as well as DEC and IBM midrange systems, and a wide array of personal computers and graphics systems.

The department currently has ongoing sponsored research in excess of \$2 million. Salary is competitive.

Applications Deadline: February 1, 1989. To apply—send resume to: Dr. James A. Senn, Chairman, Department of Computer Information Systems, Georgia State University, University Plaza, Atlanta, Georgia 30303, (404) 651-3880.

An Equal Educational and Employment Opportunity Institution.

UNIVERSITY OF CALIFORNIA, DAVIS

Faculty Positions in Electrical Engineering and Computer Science

The Department of Electrical Engineering and Computer Science at UC Davis invites applications for tenure track positions at all ranks. The primary areas of interest are image processing, computer engineering, computer science, and optoelectronics.

The department, with forty-six faculty members and 150 full-time graduate students, is experiencing rapid growth. Our College is the nation's sixteenth largest producer of engineering Ph.D.'s in a University which has the nineteenth largest extramural research funding. Salary and benefits are extremely attractive.

Davis is a pleasant, family-oriented community near Sacramento, within easy driving distance to Silicon Valley, the Lawrence Livermore National Laboratory, San Francisco, the Pacific Ocean, and the Sierra Nevada Mountains.

We are seeking individuals with strong records of teaching and research and with ambitious plans. Senior appointments require outstanding records of achievement; junior appointments must show evidence of great promise. All faculty are expected to have a strong commitment to teaching at all degree levels, and to demonstrate the ability to attract significant research support.

The positions require a Ph.D. degree or equivalent, and are open until filled. Send a resume and the names of at least three references to:

Professor S. Louis Hakimi, Chair
Attention: Faculty Search Committee
Department of Electrical Engineering and
Computer Science
University of California
Davis, CA 95616

The University of California, Davis, is an equal opportunity/affirmative action employer.

UNIVERSITY OF CALIFORNIA AT IRVINE

Faculty Positions in Computer Science

The Department of Information and Computer Science (ICS) is actively recruiting new faculty members. We have energetic research groups in the areas of architecture and operating systems, artificial intelligence and machine learning, software engineering and programming environments, social and managerial analysis of computing, and data structures and algorithms. We are currently building on these areas of existing strength.

We are looking for energetic, congenial candidates who would thrive in a serious but friendly research setting, and who would like to join us in exploring the nature of computing, broadly defined. Exceptionally distinguished candidates for senior positions are especially encouraged to apply.

The ICS Department is an independent academic unit reporting to the Executive Vice Chancellor. ICS maintains an emphasis on core computer science as well as effective interdisciplinary ties to colleagues in Neurobiology, Cognitive Science, Management, Engineering, and other areas. The

department currently has 25 full-time faculty positions and 85 Ph.D. students, with major support from the administration to expand and to strengthen the research environment. Annual research contracts from DARPA, NSF, ONR, and other agencies currently total over \$3 million. In 1986 the Department was awarded a Coordinated Experimental Research (CER) grant from the National Science Foundation. This support is fostering the creation of a Laboratory for Software Research, in which major studies of the development and evaluation of software technology can be undertaken. It is also being used to strengthen the base of machines supporting other software research. Department equipment includes over 100 workstations, primarily Sun-3's. Two large multiprocessor Sequents are available, as well as approximately 75 Macintosh Plus's and II's. Several laser printers are available for high-quality output. All machines are tied together with networks, which are gatewayed to the campus network, and from there, to regional, national, and international networks (Darpa Internet, CSnet, Bitnet, etc.). In addition, department members have access to campus-wide computing resources as well as regional supercomputer access.

UCI is located in Orange County, three miles from the Pacific Ocean near Newport Beach, and approximately halfway between Los Angeles and San Diego. The campus is situated in the heart of a national center of high-technology enterprise, and is experiencing substantial growth with exciting opportunities. Salaries and benefits are competitive. Special housing assistance is available from the university, including newly built, for-sale housing within short walking distance from the Department.

Send resumes and names of four references to:

John L. King, Chair
Department of Information
and Computer Science
University of California
Irvine, CA 92717

Apply before March 1, 1989, to ensure consideration.

An Affirmative Action/Equal Opportunity Employer.

UNIVERSITY OF OKLAHOMA

Faculty positions in electrical engineering and computer science at the University of Oklahoma. One tenure track position is currently open and others are anticipated. Ph.D. in electrical engineering or computer science required. Areas of particular interest include: computer engineering/VLSI, computer architecture, optoelectronics, signal and image processing, robotics/controls, solid state devices, and power systems. Applications will be considered at the assistant, associate and professor levels. Send vita to J.C. Thompson, Chairman of the Faculty Search Committee, School of Electrical Engineering and Computer Science, 202 W. Boyd, Rm. 219, Norman, OK 73019. First screening will begin December 9, 1988 and continue until positions are filled. One position is available on January 1989 or August 1989

UNIVERSITY OF MAINE Computer Science Department

A tenure track position is available for the Fall Semester 1989 at the level of Assistant Professor. Ph.D. in Computer Science or related discipline required. Salary and benefits are very competitive. Preference will be given to candidates with a primary interest in graphics, database systems, programming languages, software engineering or expert systems and a secondary interest in theoretical computer science. Responsibilities include teaching two courses each semester and independent or collaborative research. The department is young, has nine members, and a close relationship with other departments on campus. A graduate program was established in the Fall of 1986. Facilities include an IBM 3090 processor with a vector attachment operating under VM/370, a VAX 11/780 and DG MV/10000. The department has a multivendor workstation/PC network. The department is in the process of developing an active technology transfer program and opportunities exist for R&D work with regional industry. The University is one and one-half hours from the beautiful Bar Harbor region and Acadia National Park, and the same distance to skiing and wilderness hiking areas. Send resume and three letters of reference to: Prof. George Markowsky, Computer Science Department, University of Maine, Orono, ME 04469. Consideration of candidates will begin January 15, 1989 and preference will be given to people who apply by that time. The University of Maine is an Equal Opportunity/Affirmative Action Employer.

CALIFORNIA STATE UNIVERSITY San Bernardino

POSITION FOR ASSISTANT OR ASSOCIATE PROFESSOR (tenure track) of **COMPUTER SCIENCE**. Salary range \$33,660-\$52,068 dependent on qualifications and experience. Moving expenses. Available January, April or September 1989.

Duties include teaching, advising, curriculum development, research, and community interaction. Teaching load equated to 12 hrs/wk of lecture and lab. Ph.D. in CSci is required. Facilities include CDC, Prime, and DEC computers, many PCs, microprocessors, terminals and several LANs.

The University is located in one of the fastest growing service areas in the USA with a current student population of about 9000 and 300 computer science majors. The area is noted for its warm climate and is within a 1-2 hr drive of Palm Springs, mountain ski resorts, beaches, and metropolitan Los Angeles. Housing costs average 20% lower than the LA area. Applicants should submit letter of application and vitae. Women and minorities are encouraged to apply. Position open until filled but no later than May 15. Send materials to: Dr. Richard J. Botting, Chair, Computer Science Dept., California State University, 5500 University Parkway, San Bernardino, CA 92407, PAAAAAR@CALSTATE.BITNET

An Equal Opportunity/Affirmative Action, Section 504, Title IX Employer.

**UNIVERSITY OF CALIFORNIA,
SANTA BARBARA
Department of Computer Science**

The University of California at Santa Barbara invites applications for two tenured positions in the Department of Computer Science, starting in the 1989-90 academic year. These positions are intended for **senior persons with distinguished research records.** The Department of Computer Science is in a rapidly expanding College of Engineering. UCSB and the College of Engineering are strongly committed to supporting the Computer Science Department as it builds a Department of excellence and national visibility.

Outstanding senior persons in all areas of Computer Science will be considered, although the Department is currently attempting to achieve its main strengths in the area of software systems, parallel and distributed computing, scientific computation and machine intelligence. Senior appointees will guide the growth and development of the Department. Resources will be available for state-of-the-art laboratories for research and instruction, and strong support will be made available and tailored to the needs of successful applicants. Interactions with various research groups on campus are strongly encouraged and supported.

Applicants should hold a doctoral degree in Computer Science or a related field and must have an excellent record of research. Teaching experience is highly desirable. Deadline for receipt of applications is **February 1, 1989.** Send resume and names of referees to:

Chairman, Planning and Recruitment
Committee
Department of Computer Science
University of California
Santa Barbara, CA 93106

The University of California is an Equal Opportunity/Affirmative Action Employer. Proof of U.S. citizenship or eligibility for U.S. employment will be required prior to employment (Immigration Reform and Control Act of 1986).

**UNIVERSITY OF NEBRASKA-
LINCOLN
Computer Science and Engineering
Faculty Positions**

Applications are invited for several tenure-track positions in Computer Science and Engineering. Preference will be given to the rank of Assistant Professors. Requires a Doctorate degree in Computer Science, Computer Engineering or a related field and a strong commitment to research and teaching.

Candidates are sought from all areas of Computer Science and Engineering. Expertise in any of the following areas are especially welcome: computer networks, computer architecture, parallel processing, VLSI, operating systems, computation, analysis of algorithms, artificial intelligence, neural networks, and programming languages.

The department has in place rigorous programs in computer science and engineering research and education, and is expanding in the direction of computer engineering. A

Center for Communication and Information Science research has been established to support departmental research. Industry ties with the center have been established and external funding is sought through the center. The department offers degrees in two colleges, Arts and Sciences and Engineering and Technology.

The University of Nebraska-Lincoln is the primary campus for research and graduate studies in Nebraska. UNL provides a modern computing environment to all its faculty, and computer science and engineering faculty have a local area network and access from their offices to departmental and campus computer and to CSNET. Currently about 37 Master's and 18 Ph.D. candidates are enrolled in computer science graduate programs.

The Department of Computer Science and Engineering has 15 full-time faculty with research interests in analysis of algorithms, artificial intelligence, computer architecture, fault-tolerant computing, VLSI, data bases, programming languages, formal languages, combinatorics, numerical analysis, symbolic and algebraic computation, neural network simulation, computer networks, theory of computation, parallel processing, computer vision, performance evaluation, and human-computer interaction.

Applicants should submit vitae and names of three references to:

Chairman, Search Committee
Computer Science and Engineering
115 Ferguson Hall
University of Nebraska-Lincoln
Lincoln, Nebraska 68588-0115

Applications should be received by January 6, 1989; however, applications will continue to be accepted until all positions are filled. Applications from women and minorities are especially encouraged.

Affirmative Action/Equal Opportunity Employer.

UNIVERSITY OF MARYLAND

Computer Vision/Human-Computer Interaction: Assistant, Associate and Full Research Scientists to direct advanced research programs in computer vision or human-computer interaction. One year renewable positions whose availability and continuation are dependent on contract/grant funds. Depending on the area of research, Ph.D. in computer science, psychology, or closely related discipline required. For computer vision positions, experience in computer vision research required, specifically in areas such as three-dimensional and time-varying scene analysis, robot navigation, knowledge based vision systems, and computer architectures for vision. For human-computer interaction positions, experience is required in areas such as development of user interfaces and human factors testing. Salary up to \$60,000 depending on level of position. Closing date for accepting applications is September 1, 1989. Indicate which position and for which area of research you are applying and submit signed and dated curriculum vitae plus three references to: Barbara Hope, Center for Automation Research, University of Maryland, College Park, MD 20742-3411. AA/EOE.

**LOUISIANA STATE UNIVERSITY
Computer Faculty Positions**

The Department of Electrical and Computer Engineering at LSU invites applications for anticipated tenure-track and visiting faculty positions available August 1989 in all areas of computer engineering, including microprocessors, distributed processing systems and special purpose architectures. A Ph.D. or equivalent and potential for excellence in teaching and research are necessary. Rank is open. Salary is competitive and commensurate with qualifications and experience. Release time and resources are provided in order to enhance the development of a quality research program. Opportunities for summer support are available. Send resume, names of three references, a statement of teaching and research interests, and verification of employment eligibility in compliance with the Immigration Reform and Control Act of 1986 to: Alan H. Marshak, Chairman, Electrical and Computer Engineering, Louisiana State University, Baton Rouge, LA 70803-5901. LSU is an Equal Opportunity Employer.

**DIRECTOR
SYSTEMS RESEARCH CENTER
VIRGINIA POLYTECHNIC
INSTITUTE & STATE UNIVERSITY**

Virginia Polytechnic Institute & State University (Virginia Tech) invites applications and nominations for the position of Director of the Systems Research Center. The Center is a University entity specializing in interdisciplinary research and development related to time-critical systems in general, and naval real-time embedded systems in particular. Largely funded by the U.S. Navy, Center projects have focused on computing technology, and its broad impact on Navy systems. Applicants should demonstrate an outstanding record of research accomplishments commensurate with a senior-level appointment in an academic department affiliated with the Center: Computer Science or a department in the College of Engineering. A teaching and research specialization in digital systems technology is highly desirable as well as in-depth knowledge of the Navy research, engineering, and systems communities. Applicants should possess strong managerial skills, have a record of interdisciplinary research experience, and exhibit the ability to communicate well with Navy and University personnel.

Applications and nominations will be received until the position is filled. Applicant screening will begin December 1. Applications should include a statement of research goals, curriculum vitae, and names and telephone numbers of three references. Send to:

E.R. Stout
Associate Provost for Research
Research Division
Virginia Tech
Blacksburg, VA 24061-0244

Virginia Tech is an Equal Opportunity/Affirmative Action employer.

**ROCHESTER INSTITUTE
OF TECHNOLOGY**
Director
School of Computer Science

Nominations and applications are invited for the position of Director, School of Computer Science. RIT is a privately endowed, coeducational, comprehensive university, located on 1300 acres in suburban Rochester, NY, with an enrollment of 13,000 students in nine colleges. The School of Computer Science is administratively housed in the College of Applied Science and Technology. The emphasis in most RIT degree programs is on career education, with students actively involved in cooperative education.

The Director is responsible for academic policy and quality in the school, and has significant authority in the allocation of resources among the school's teaching and research programs. The school consists of three departments: Applied Computer Studies, Graduate Computer Science, and Undergraduate Computer Science. There are a total of 30 faculty members in the three departments, 550 undergraduate students and 250 graduate students. The school offers B.S. and M.S. degrees in Computer Science, an M.S. degree in Software Development and Management, and an Advanced Certificate in Applied Computer Studies.

Candidates must be capable of leadership in an academic environment and possess strong fiscal and organizational management skills. They must show evidence of significant administrative experience and the ability to work with faculty from diverse academic programs. An earned doctorate in a relevant discipline is preferred.

Send letter of application, describing educational philosophy and management style, curriculum vitae, and a list of three references, by January 15, 1989 to Prof. Guy Johnson, Chairman, Director Search Committee, School of Computer Science, Rochester Institute of Technology, Rochester, NY, 14623. RIT is an AA/EO employer.

**UNIVERSITY OF
WISCONSIN-MADISON**
Faculty Positions

The Department of Electrical and Computer Engineering invites applications for tenure and tenure-track positions. A Ph.D. degree is required, and successful candidates are expected to participate in both teaching and research activities. Applicants in all areas of computer engineering are invited to apply, but the following areas are of special interest: computer architecture, computer networks, VLSI and computer-aided design, microprocessor and minicomputer applications, real-time control and instrumentation applications, and engineering applications of artificial intelligence. Rank and salary will be commensurate with qualifications and experience. Send resume and names of three references to J. Leon Shohet, Chairman, Department of Electrical and Computer Engineering, University of Wisconsin-Madison, 1415 Johnson Drive, Madison, WI 53706, an equal opportunity/affirmative action employer.

**UNIVERSITY OF CALIFORNIA at
SANTA CRUZ**
Baskin Center for
Computer Engineering and
Information Sciences

The Computer Engineering Board of Studies at the University of California's Santa Cruz campus invites applications for positions of Assistant, Associate or Full Professor, as appropriate, effective July 1, 1989. Application closing date is January 18, 1989.

The Division of Natural Sciences at UCSC, through the Baskin Center for Computer Engineering and Information Sciences, offers programs in Computer and Information Sciences and in Computer Engineering. Research is emphasized in seven different areas: theoretical computer science, artificial intelligence, programming languages and environments, computer graphics and image processing, communications and networks, computer architecture and operating systems, and computer systems design and applications. These fields of emphasis are supported by ten faculty members from the Computer Engineering board and fourteen faculty members from the Computer and Information Sciences board.

UCSC is the University of California campus nearest to "Silicon Valley". Faculty salaries are competitive, and opportunities for consulting are extensive.

Candidates are sought at the Assistant Professor level with strengths in two or more of the following areas: special purpose architectures and parallel environments, computer aided design, graphics, real time systems, image processing, computer architecture, computer systems and languages, and software construction. (#142-889)

Candidates are sought at all levels, with preference given to senior level applicants, with strengths in two or more of the following areas: software engineering, compilers, computer architecture, operating systems, algorithms, workstation networking, parallel environments, robotics, image systems, and VLSI (#141-889)

Minimum qualifications: Ph.D. in Computer Engineering, Electrical Engineering, Computer Sciences, or equivalent. Candidates for tenured positions must have a solid research record as evidenced by publications in technical journals. Candidates for non-tenured positions must have demonstrated potential for a successful research career. Applicants will be evaluated on their research record, teaching, professional activities, and demonstrated (or potential) leadership in their fields. Industrial experience will be favorably considered.

Send applications, including curriculum vitae with cover letter, letters of recommendation, copies of any recent publications, and teaching evaluations, to the address listed below. Candidates for tenured appointments should obtain at least five letters of recommendation evaluating their scholarly contributions, teaching, and other professional accomplishments. Candidates for Assistant Professor appointments should obtain at least three letters of recommendation.

Applications should be postmarked by January 18, 1989. If either position remains unfilled, applications at the Assistant Pro-

fessor level only will be accepted until February 28, 1989. If positions at this level remain unfilled, consideration may be given to junior level applications received until May 1, 1989.

Salary commensurate with qualifications and experience.

Send applications to: Chair, Computer Faculty Search Committee, Baskin Center for Computer Engineering and Information Sciences, Applied Sciences Building, University of California, Santa Cruz, CA 95064. Please refer to positions by field: #141-889 or #142-889. Please indicate whether applications for 141-889 are at the junior or senior level.

UCSC IS AN EEO/AA/IRCA EMPLOYER.

**THE UNIVERSITY OF TEXAS
AT ARLINGTON**

The Department of Computer Science Engineering at The University of Texas at Arlington invites your application for tenure-track or visiting faculty positions in all areas of computer science and computer engineering. Rank is open. An earned doctorate or equivalent and commitment to teaching and scholarly research is required. Openings are expected for September 1989. Applications received prior to March 1, 1989 will receive full consideration. Interested persons should send a resume to Bill D. Carroll, Professor and Chairperson, Computer Science Engineering Department, P.O. Box 19015, The University of Texas at Arlington, Arlington, TX 76019. Phone 817-273-3785.

The University of Texas at Arlington is an Equal Opportunity/Affirmative Action Employer.

PENN STATE
Computer Engineering

Applications are invited for faculty positions at all levels. Candidates from all areas of computer engineering (hardware and software) will be considered. The Computer Engineering Program at The Pennsylvania State University is within the Department of Electrical Engineering which has over 50 faculty members, and approximately 1300 undergraduate majors, 200 graduate students. Candidates should have a Ph.D. in Electrical/Computer Engineering or related areas. There are 13 faculty members within the Computer Engineering Program. Excellent instruction and research computing facilities are available within the Department, College and at the University Computation Center.

Send letter of application, resume, or inquiries, together with three references to: COMPUTER ENGINEERING PROGRAM, DEPARTMENT OF ELECTRICAL ENGINEERING, 129 ELECTRICAL ENGINEERING EAST Box-C, THE PENNSYLVANIA STATE UNIVERSITY, UNIVERSITY PARK, PA 16802.

Deadline for applications is November 30, 1988, or until suitable qualified candidates are selected.

An Affirmative Action/Equal Opportunity Employer Women and Minorities Encouraged to Apply.

STATE UNIVERSITY OF NEW YORK AT BUFFALO

The Department of Computer Science is seeking to fill several Assistant Professor positions to begin Fall 1989. Applicants must have a Ph.D. in Computer Science or a related field by September 1, 1989, and must have superior research ability.

The University at Buffalo is the largest public university in New York and New England, with approximately 26,000 students enrolled in 93 undergraduate, 101 master's, and 94 doctoral programs, including law, medical, and business schools.

Present computer science research areas include artificial intelligence, computer vision, parallel algorithms, theory of computation, and performance evaluation. The annual research expenditure of the department exceeds \$1 million. We are particularly interested in faculty who will be able to bridge the areas of present strength to the systems areas of programming languages, computer architecture and operating systems. The department offers BA, BS, MS and PhD degrees, with controlled undergraduate enrollment. Departmental computing facilities include a VAX 11/785, twenty SUNs, nine Lisp machines, two Hypercubes, one Encore multimax and several image processing/graphics systems. The current department consists of 15 tenure-track faculty, 3 lecturers, 3 full-time technical staff and over 60 supported graduate students. The department is expanding, with additional faculty lines committed annually. Salaries are extremely competitive.

Resumes and names of four references should be directed to: Prof. Sargur N. Srihari, Chairman of Search Committee, Department of Computer Science, 226 Bell Hall, SUNY at Buffalo, Buffalo, NY 14260 (CSnet: srihari@buffalo). For full consideration, applications should be received by January 15, 1989. SUNY is an affirmative action/equal opportunity employer.

Numerical Aerodynamic Simulation Systems Division NASA Ames Research Center

The Numerical Aerodynamic Simulation (NAS) Systems Division is seeking experienced computer professionals at the senior level for challenging positions in large scale computer systems research, development and operational support at the NASA Ames Research Center located on Moffett Field, CA.

The NAS Facility houses one of the world's largest and most advanced supercomputer systems, used by a nationwide user community for solving highly complex problems in fluid dynamics, aeronautics and other scientific disciplines. The system includes a CRAY Y-MP, a CRAY-2, a dual Amdahl 5880's, four VAX 11/780's, 30 Silicon Graphics Iris Workstations interconnected with Local Area Networks ranging in speeds from 10 to 800 mbits per second. The user visible operating system is UNIX with TCP/IP based networking extensions.

System research, development, project management and computational services opportunities exist in the following areas:

- 1) Supercomputers
- 2) Mass Storage Systems
- 3) High Performance Scientific Workstations
- 4) High Performance Data Networks
- 5) Long Haul Communications
- 6) Mini-supercomputers

These positions require at least a Bachelor's degree in computer science, engineering or one of the physical sciences. A minimum of three years directly related experience in the development of large software and hardware systems is required. Knowledge of the UNIX operating system and the DoD Internet Protocols is highly desirable. Salary (GS 13/14 \$39,501-\$60,683). The NASA Ames Research Center is an Equal Opportunity Employer. U.S. Citizenship is required. Resumes should be sent to:

Mr. Bruce Blaylock
Chief, NAS Systems Development
Branch
M/S 258-5 [B-2]
NASA Ames Research Center
Moffett Field, CA 94035

COMPUTER SYSTEMS ANALYST

Will accept entry level with B.S. degree Computer Science.

Will be required to analyze our banking problems such as deposit/withdrawal control, loan processing, retirement investment, outstanding loans, real estate investment, customer relations, etc. to refine our system and convert it to programmable form for application to electronic data computer systems. Will consult with top management and department heads to ascertain specific output requirements such as degree of data summarization and format for management reports. Will also develop process flow charts and diagrams and audit trail printouts. Salary \$37,700 year. Send resume and this ad to Chun Kyu Lee, The Wilshire Bancorporation, 455 Wilshire Blvd., Los Angeles, California 90010.

THE UNIVERSITY OF TEXAS AT AUSTIN

Electrical and Computer Engineering Department is accepting applications for tenure and tenure-track faculty positions. Qualifications must be commensurate with rank and include an outstanding academic record, significant achievement in original research, sincere interest in teaching, and a doctorate in electrical or computer engineering/science. Duties will include teaching undergraduate and graduate courses. Interested applicants are invited to send resumes which detail their past professional accomplishments, and which include the names and addresses of three references, to Dr. Edward J. Powers, Chairman, Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, Texas 78712-1084, an Equal Opportunity/Affirmative Action Employer.

THE WICHITA STATE UNIVERSITY Chairperson Computer Science Department

Applications are invited for the position of Chairperson, Department of Computer Science, The Wichita State University, an urban institution enrolling over 17,000 students. The Department, with ten full time professional positions and eight instructors and support staff, is housed in the College of Liberal Arts and Sciences. It serves approximately 500 undergraduate majors and 120 graduate students, and currently offers BA, BS, MCS, and MS degrees.

Wichita is located in the heart of the aviation industry, and the University is committed to basic and applied research. There are ample opportunities for contract research with industry. Departmental equipment includes several VAX computers, a Symbolics 3670, and four Ada workstations. The department is the first graduate curriculum test site for the Software Engineering Institute. Faculty strengths include artificial intelligence, software engineering, image processing, data bases, communications networks, and the mathematical foundations of computer science.

Applicants must hold the Ph.D. in Computer Science or a related discipline; experience in academic positions, an established record of research, teaching, service, and administrative ability are required. Rank and salary are dependent on qualifications and experience.

A resume and names of three references should be submitted by January 31, 1989 to Dr. James E. Tomajak, Department of Computer Science, Wichita State University, Wichita, KS 67208; phone: (316) 689-3156; net address jct@sef.cmu.edu. Wichita State University is an Equal Opportunity/Affirmative Action Employer.

NEW MEXICO TECH Computer Science Faculty Position

Applications are invited for a tenure-track position in Computer Science. The level is open and the salary is competitive. Candidates must have a PhD in Computer Science and demonstrate potential for excellence in teaching and research. Areas of particular interest are operating systems and computer networks. Duties include teaching graduate and undergraduate classes, research and thesis supervision, advising, and service to the department. New Mexico Tech is a scientific and technical institution with 1200 students. The CS department (offering BS, MS, and PhD) has 130 talented students. There are excellent facilities; and the department is planning to further upgrade its equipment. Nearby institutions include the National Radio Astronomy Observatory, Los Alamos and Sandia National Laboratories, and computer companies with facilities in Albuquerque. Tech is located in the Rio Grande valley with fabulous weather and endless outdoor recreational opportunities. Send applications to: Chairman, Computer Science Search Committee, Personnel Office, New Mexico Tech, Campus Station Box C-97, Socorro, NM 87801. AAEOE.

SANTA CLARA UNIVERSITY
Tenure Track Position In
Computer Engineering

Santa Clara University's EECS department has an immediate opening for a tenure track faculty position in Computer Engineering. Candidates with specialization in Computer Architecture or Software Engineering.

Applicants must have a Ph.D. in Computer Engineering or a closely related area with a commitment to both teaching and research. Senior applicants must have a distinguished research and teaching record.

SCU is located 45 miles south of San Francisco, in the center of Silicon Valley. The department offers B.S., M.S., and Ph.D. degrees in both computer and Electrical Engineering. Engineering computing facilities include more than 70 engineering workstations, besides a VAX-11/750. University academic computing resources include a VAX-8650 plus more than 800 IBM PCs. Most computing facilities are networked together over a campus wide local area network.

Candidates should submit a resume including the names of at least three references to:

Dr. Hasan S. Alkhatib, Chairman

Faculty Search Committee

EECS Department

Santa Clara University

Santa Clara, California 95053

E-mail: HALKHATIB@SCU.BITNET

Santa Clara University is an Affirmative Action/Equal Opportunity Employer.

UNIVERSITY OF
SOUTHWESTERN LOUISIANA
The Center for
Advanced Computer Studies
Research Faculty
Teaching Faculty
Graduate Fellowships/Assistantships
in Computer Science/Engineering

The Center for Advanced Computer Studies is a research center with programs leading to MS/PhD degrees in Computer Science and Computer Engineering. External grants/contracts support research in a variety of areas. The Computing Research Laboratory includes a 40-node Sun-3 network, an Encore parallel processing system, 2 VAX 11/780s, a comprehensive digital design lab, laser printers, plotters, and other equipment. Instruction utilizes a 3-processor Pyramid 90X network running UNIX and an IBM 3090-200 with a vector processor.

Another well-equipped facility supports research in CAD/CAM. About 300 students are enrolled in computing graduate programs, including 120 for the PhD. The undergraduate program in the Computer Science Department is accredited by CSAB and offers both scientific and commercial options, with a current enrollment of 450. The undergraduate program in the Electrical and Computer Engineering Department is accredited by ABET and offers an option in Computer Engineering, with a current enrollment of 212.

RESEARCH FACULTY: Applications are invited from persons holding PhD degrees with demonstrated research capabilities in Computer Science/Engineering. Associate Professors and Professors must hold PhDs and have an established research publication and grant record. Persons appointed as Assistant Professors must hold PhDs in Computer Science/Engineering and have research potential. The typical teaching load is 2 graduate-level courses per year and a continuing research seminar. Salaries are competitive and excellent support for travel, equipment, research assistants, and professional activities is provided so you can achieve your professional goals. To apply, send a copy of your resume and the names and addresses of at least 3 professional references. Applications will be considered until all positions are filled.

TEACHING FACULTY: Tenure track positions are available for persons holding PhD degrees in Computer Science or Computer Engineering with a strong commitment to teaching undergraduates. Demonstrated interest and potential for performing limited independent research is desirable. Duties include teaching, advising, curriculum development, and program evaluation. Competitive salaries, support for travel, well equipped laboratories, and graduate assistant support provide an environment for quality instruction. To apply, send a copy of your resume and the names and addresses of at least 3 professional references. Applications will be considered until all positions are filled.

PHD FELLOWSHIPS: A number of PhD Fellowships valued at up to \$18,000 per year including tuition and fees are available. They provide support for up to 4 years of study towards the PhD in Computer Science or Computer Engineering. Recipients also receive preference for low-cost

campus housing. Applications must be received by 15 February 1989.

MS/PHD ASSISTANTSHIPS: More than 100 teaching and research assistantships are available to support students pursuing an MS or PhD degree in Computer Science or Computer Engineering. Stipends are valued at up to \$10,000 per academic year including tuition and fees. Applications must be received by 1 March 1989.

APPLICATIONS: Dr. Steve P. Landry, Acting Director, The Center for Advanced Computer Studies, Lafayette, LA 70504-4330. Phone: (318) 231-6284.

An Affirmative Action/Equal Opportunity Employer.

SOUTHERN ILLINOIS UNIVERSITY
AT CARBONDALE

Applications are invited for two (2) tenure track faculty openings in a growing dynamic Computer Science Department. Candidates in all fields of computer science or computer engineering will be considered. Evidence of ongoing and future research, a commitment to teaching, and willingness to participate fully in the department's graduate and undergraduate programs are basic requirements. Applicants should have, or expect to receive in 1989, a Ph.D. in computer science or a closely related discipline. Departmental facilities include a network of personal computers, VAX/Sun/Xerox workstations, and a shared memory parallel machine—the Sequant Balance 8000. In addition, a host of IBM mainframe computers are available for research and teaching purposes. The positions are at the assistant/associate level. One position has a starting date of January 1, or August 16, 1989. The second position is to begin August 16, 1989. To be considered for a January appointment, applications must be received by November 15, 1988. For an August appointment, applications will be accepted until March 1, 1989, or until the positions are filled. Resumes should be sent to: Faculty Recruitment Committee, Department of Computer Science, Southern Illinois University, Carbondale, IL 62901-4511. SIUC is an Equal Opportunity, Affirmative Action Employer.

ROCHESTER INSTITUTE
OF TECHNOLOGY
School of Computer Science

Visiting tenure track faculty position available in September or December for individual who has experience in the instruction of Computer Science or Software Engineering (preferred), Ph.D. or Master's (with professional experience) degree in appropriate discipline required. Duties will include teaching at the undergraduate and graduate (M.S.) level as well as applied research (Software Engineering preferred).

Send resume to Evelyn Rozanski, Director, School of Computer Science, 1 Lomb Memorial Drive, Rochester, New York, 14623. RIT is an equal opportunity affirmative action employer.

UNIVERSITY OF MINNESOTA
Supervisory Applications Programmer,
Class #1454
Job Requisition No. 004267

The Natural Resources Research Institute at UMD has an immediate opening for a Supervisory Applications Programmer.

Qualifications: BS or BA degree in computer science; programming experience in C, Fortran, and Pascal; 5 years experience in QSAR research; and 5 years of supervisory experience.

Description: The Supervisory Applications Programmer will join the research staff in the administration, development, and supervision of software and data bases to support QSAR research. Desirable experience includes publications in peer-reviewed journals and a demonstrated record in obtaining extramural funding in QSAR research.

Salary: \$2278 to \$3753 Monthly. **Starting date:** January 1, 1989 or as soon thereafter as possible. To apply, please send a University of Minnesota application form; current resume including publications and summary of extramural funding; and name, phone, and address of 3 references to: Employment Office; University of Minnesota, Duluth; 281 Darland Administration Building; Duluth; MN 55812. Applications must be postmarked by December 1, 1988. THE UNIVERSITY OF MINNESOTA IS AN EQUAL OPPORTUNITY EMPLOYER AND EMPLOYER.

WORCESTER POLYTECHNIC INSTITUTE

The Computer Science Department invites applications for tenure track faculty positions at all levels. Candidates from all areas of computer science will be considered; however preference will be given to candidates with expertise in theoretical aspects of computer science and computer architecture. Candidates should have a Ph.D. in Computer Science and a strong interest in both research and teaching.

Worcester Polytechnic Institute emphasizes quality in the undergraduate learning experience and is committed to an innovative project-oriented teaching environment. The undergraduate computer science degree is accredited by the Computer Sciences Accreditation Board.

The current goal of the Institute is to enhance our graduate program and improve research activities. The department seeks qualified candidates who will help us achieve these objectives. WPI is located close to the center of the Massachusetts minicomputer industry and excellent opportunities exist for cooperative research and consulting.

The Department has 12 full-time faculty with 170 undergraduates and 50 full-time and 100 part-time graduate students in our M.S. and Ph.D. programs. Department equipment includes an Encore MultiMax, four VAX 750s, an AT&T 3B-15, seven Sun workstations, and more than 80 PCs (both UNIX and MS-DOS based). Most of this equipment is interconnected via Ethernet. The Institute is completing the installation of a campus-wide telecommunication system which will provide complete connectivity throughout the campus. A new Information Science Building is currently under construction, and we plan to move into our new facilities during the 1989 fall semester.

Located 45 miles west of Boston, Worcester is a city of 180,000 people. There are seven colleges and universities in greater Worcester, and Worcester offers a rich variety of cultural activities.

Please send a resume to Robert Kinicki, Department Head, Department of Computer Science, WPI, Worcester, MA 01609.

WPI is an Equal Opportunity/Affirmative Action Employer.

UNIVERSITY OF MINNESOTA

Department of Electrical Engineering has several tenure track or tenured faculty positions in Electrical Engineering available at all levels. The duties will include the establishment of sponsored research and undergraduate and graduate teaching. Applications from individuals with exceptional qualifications in all areas of Electrical Engineering will receive due consideration. Of particular interest, however, are specialties in computer architecture and digital electronics, microelectronics, circuits and VLSI design, magnetism, image processing and computer vision.

The Department of Electrical Engineering has 45 faculty members providing undergraduate and graduate education and pursuing research in all areas of electrical engineering. The State-funded Microelectronics and

Information Sciences Center and Supercomputer Institute at the University of Minnesota will provide opportunities for interdisciplinary research in cooperation with industry. The Department has a complete integrated circuit fabrication and design lab, access to Cray supercomputers, and other facilities.

The requirements include an earned Doctorate and outstanding academic and research records. Rank and salary will be commensurate with qualifications and experience. Temporary and visiting positions are also available. Send applications and resumes to:

Professor Mostafa Kaveh, Chairman of the Faculty Recruiting Committee, Department of Electrical Engineering, University of Minnesota, 200 Union Street Southeast, Minneapolis, MN 55455.

Last date for receiving applications August 30, 1989, for positions available September 16, 1989. Review and interviews may begin as early as November 15, 1988, and early decisions may be made on some positions.

The University of Minnesota is an equal opportunity educator and employer and specifically invites and encourages applications from women and minorities.

PURDUE UNIVERSITY CALUMET Department of Computing, Telecommunications and User Services

Urban university, southeast of Chicago, invites qualified candidates to submit applications for positions in an expanding computer and telecommunications department. Campus computer resources currently include an IBM 4341 mainframe, three clustered Digital VAX computers, a Four Phase IV/95 word processing system, and over 400 networked personal computers.

SYSTEMS PROGRAMMER — Two Positions

Responsible for installation and maintenance of the IBM VM/SP and DOS/VSE or Digital VAX/VMS operating systems and related subsystems. Bachelor's degree in computer technology, computer science, or other disciplines considered based on work experience required. Two years systems programming work experience required (IBM mainframe and/or Digital equipment operating systems preferred). Working knowledge of VMS or VM/SP and DOS/VSE required.

Both positions require good written and oral skills. Applications for positions will be accepted through November 14, 1988. Minority and female candidates are encouraged to apply.

Send cover letter, resume, and academic transcripts to:

Robert Hopper, Associate Director
Computing, Telecommunications and
User Services

PURDUE UNIVERSITY CALUMET
Hammond, IN 46323-2094

An Affirmative Action/Equal Opportunity Employer.

GEORGIA INSTITUTE OF TECHNOLOGY School of Information and Computer Science

The School of Information and Computer Science invites applications for faculty positions at all levels. Applicants should have a commitment to teaching and a record of outstanding research accomplishments. Applicants who expect to receive a Ph.D. degree by Fall 1989 and who show high potential for research as well as a commitment to teaching are also invited.

The School seeks applicants to strengthen its capabilities in all areas of computer science. Very competitive salaries are offered.

The School has 30 faculty members and anticipates further faculty growth. Its educational activities include an undergraduate program accredited by the Computing Sciences Accreditation Board, Inc., a Masters program with 150 students and a Ph.D. program with over 70 students. Well equipped laboratories support research and education. High-speed local area networks interconnect all major campus laboratories and provide access to national networks.

The School is in the second year of a five year Coordinated Experimental Research grant from the National Science Foundation. This grant is funding acquisition of hardware and development of software to support experimental work in parallel and distributed computing.

Georgia Tech is located in Atlanta, which experiences a mild sunbelt climate. It is the center of commerce in the Southeast, offering a diverse economy and good employment opportunities in all professional areas. Atlanta offers good cultural and recreational opportunities, extremely attractive residential neighborhoods, and affordable housing.

Candidates should send complete resumes and names of at least three references to: Chairman, Faculty Search Committee; School of Information and Computer Science; Georgia Institute of Technology; Atlanta, Georgia 30332.

Georgia Tech is an equal opportunity employer.

SOUTHEAST MISSOURI STATE UNIVERSITY

The Department of Computer Science invites applications for a tenure track position at the associate or full professor level. Duties include teaching 6-9 hours of undergraduate computer science courses and participation in ACM accreditation effort, curriculum development, and research coordination. Ph.D. in Computer Science, Information Systems or closely related area. Please send resume with names of three references and transcripts of all college credit to: Dr. Bill Weber, Chairperson, Department of Computer Science, Southeast Missouri State University, Cape Girardeau, MO 63701. Application deadline: February 1, 1989 or until filled.

Southeast Missouri State University is an equal opportunity M/F/affirmative action Employer.

DUKE UNIVERSITY
Department of Computer Science

The Duke University Department of Computer Science is recruiting junior faculty in theoretical computer science and in biomedical scientific computing. Applicants must demonstrate excellence in research or exhibit the promise of excellence.

The Department currently has seventeen tenure track faculty, approximately 200 undergraduate majors and 70 graduate students pursuing master's and/or doctoral degrees.

The Department has major research efforts in scientific computing with emphasis on numerical linear algebra, the solution of PDEs, and VLSI simulation; computer systems with emphasis on computer architectures, modeling of fault-tolerant systems, systems performance, and communications; artificial intelligence, particularly in the areas of natural language interface, search methodologies, and expert systems; and theory and algorithms with emphasis on combinatorial and graph-theoretic studies. Special motivation for the research efforts comes from the areas of medical applications (in collaboration with the Duke Medical Center), and VLSI (in collaboration with the Microelectronics Center of North Carolina, of which Duke is a Participating Institution).

Interested applicants should send copies of their resumes and other supporting material by February 14, 1989, to:

Professor Donald J. Rose
Department of Computer Science
Duke University
Durham, NC 27706

Duke University is an affirmative action, equal opportunity employer.

THE UNIVERSITY OF MICHIGAN
ANN ARBOR, MICHIGAN
Faculty positions available in
electrical engineering and
computer science.

Applications are solicited for faculty positions at all ranks in various areas of electrical engineering and computer science and engineering. Qualifications include an outstanding academic record, significant involvement in research, a doctorate in electrical or computer engineering or computer science, and a strong commitment to teaching and research. Areas of specialization being considered are artificial intelligence, circuits and microelectronics, communication networks, computer architecture, image processing, languages and compilers, millimeter-wave antennas and remote sensing, operating systems, optics, robotics and integrated manufacturing, software engineering, solid-state devices, theory of computer science, VLSI.

Please send resumes to:

Edward S. Davidson, Chairman
Department of Electrical Engineering
and Computer Science

The University of Michigan
Ann Arbor, MI 48109-2122

An Equal Opportunity/Affirmative Action Employer.

FLORIDA STATE UNIVERSITY
Department of Computer Science

Applications are invited for tenure track faculty positions at all levels in the Department of Computer Science. Areas of special interest include, but are not limited to, operating systems, programming languages, computer graphics, software engineering, and computer architecture.

Our department is a young and rapidly growing one. It offers B.S., M.S., and Ph.D. degrees, and it places a strong emphasis on excellence in research and teaching. Research facilities include a supercomputer ETA 10, a CYBER 205, a CYBER 850, a graphics lab, and a local network consisting of a VAX 11/780, a VAX 11/750, several SUN workstations, and Symbolics 3640.

Applicants should have a Ph.D. in computer science or a closely related area, and a strong commitment to teaching and research.

Send resume and names of at least three professional references to:

Dr. Abe Kandel, Chairman
Department of Computer Science
Florida State University
Tallahassee, FL 32306-4019

Florida State University is an Equal Opportunity Affirmative Action Employer.

TULANE

Tulane is a major private university, selective in enrollment and comprehensive in scope, with about 10,000 undergraduate and graduate students. The campus is located in a residential area of uptown New Orleans, one of America's most distinctive cities. Faculty benefits include dependent tuition, mortgage assistance, relocation expenses, 12% TIAA and typical insurance benefits.

The department computer network includes a Pyramid 9810 running UNIX, a VAX 11/780 running VMS, and several Sun workstations. This local area network is a subset of the Tulane University network which is connected to SURANET, a regional subset of NSFNET. Other major computing resources on the Tulane University network and accessible from Computer Science Offices include an IBM 3081KX mainframe. In addition, an IBM 4341 supports expert system research and instruction.

The department invites applications for faculty positions at all levels for the spring, 1989. Candidates should have a Ph.D. in Computer Science or a related area. Research excellence or demonstrated research potential and a commitment to quality teaching are required. Applicants for instructorships must possess a masters in Computer Science or ABD.

Applications should be directed to Dr. Johnette Hassell, Head, Department of Computer Science, School of Engineering, Tulane University, New Orleans, LA 70118. Applications must be received by December 1, 1988 for spring appointment. Tulane University is an Affirmative Action Equal Opportunity Employer.

ARIZONA STATE UNIVERSITY
Computer Science
Computer Engineering

The Department of Computer Science seeks outstanding faculty candidates for research and teaching in all areas of computer science and computer engineering. Applicants should have completed or be completing a Ph.D. in computer science, computer engineering, or a closely related field, and must show promise of excellence in teaching and research. The positions are all tenure track; rank is open.

Both undergraduate and graduate programs through the Ph.D. are offered. The department is an active participant in an impressive alliance of high-tech businesses, the State, and the University building an outstanding faculty and facilities for Computer Science and Computer Engineering at ASU. Although the present facilities are only six years old, the department has outgrown them, and is slated to move into new facilities scheduled for completion late in 1990.

Faculty engineering workstations are networked locally to DEC, IBM, Harris, Convex, and Honeywell mainframes, and through the campus broadband network to the university Cray XMP/14se and IBM 3090/500E supercomputers. A major part of freshman/sophomore instruction is hosted on Macintosh's, Intel 286/310 Xenix systems, and IBM PC's.

Please send a resume and the names of three references to:

Dr. Ben M. Huey,
Faculty Search Committee
Department of Computer Science
Arizona State University
Tempe, Arizona 85287-5406.

CSNET: huey@asu.edu

Deadline: January 31, 1989 or until filled. ASU is an equal opportunity/affirmative action employer.

**INDIANA UNIVERSITY-
PURDUE UNIVERSITY**
at Indianapolis

The Department of Computer & Information Science invites applications for several tenure track positions. A Ph.D. in Computer Science (or the equivalent) and strong teaching and research credentials are required. The Department is expanding and will consider strong candidates in all areas of Computer Science. Rank and salary are competitive.

The University is rapidly growing and now has over 24,000 students. Indianapolis is a large, active metropolis and has available a dynamic industrial base conducive to joint research and consulting with our faculty. The six-hospital Medical Center at IUPUI is yet another source of cross-disciplinary research for the Department. Outstanding computing facilities include a network of VAX, IBM, CDC, and Prime machines. Please send resumes and references to Dr. Michael A. Penna, Chairman Search and Screen Committee, Department of Computer & Information Science, Box 647, IUPUI, Indianapolis, IN 46223. IUPUI is an Equal Opportunity/Affirmative Action Employer.

BOOK REVIEWS

Editor, Wiley McKinzie, School of Computer Science and Technology, Rochester Institute of Technology, Rochester, NY 14623; Compmail, w.mckinzie; CSnet, wrm@rit

Database Design Fundamentals: A Structured Introduction to Databases and a Structured Application Design Methodology

Naphtali Rishe (Prentice Hall, Englewood Cliffs, N.J., 1988, 421 pp., \$40)

This is not a book on database design. It mainly concerns database programming languages, or rather, the semantics of a database programming language based on Pascal and a declarative language based on predicate calculus. The author uses the two languages in conjunction with a primitive semantic binary model. The Pascal-based language is enhanced for, and applied to, three logical database models: relational, network, and hierarchical.

Nothing is basically wrong with this book, except that it does not fill any useful niche. Its potential readership is not clear. In the preface, the author states he intends the book as

- a text for an undergraduate course on databases;
- a text for a college course for future systems analysts and programmers;
- a text for an "updating" course for software engineers, systems analysts, and programmers;
- a text for a self-educating professional;
- a supplement for a graduate course on databases; and
- a quick-reference manual and glossary of database terminology.

I do not think this book satisfies any of these purposes. In particular, readers in the first three categories could not be attracted to this book at all.

Regarding the first purpose, an introductory undergraduate book on databases should explicitly relate to popular database models and database management systems. Such a book should adhere to standards and contain enough technical details to serve as a basic reference to whatever DBMS is chosen as an implementation arena for the students. In other words, the book must be pragmatic and cover a wide range of topics.

Rishe's text is not pragmatic, although he claims it is. It does not refer to any

popular database models or DBMSs. It does not mention the most widely used semantic (conceptual) models, such as Chen's entity-relationship, Nijssen's information analysis method (NIAM), or the infological models of Langefors and Bubenko. The book describes a variation of the semantic binary model but does not reference original works by Abrial, Senko, and others. On the logical level, the adherence to standards (Structured Query Language and Network Database Language) and the depth of discussion are not adequate for undergraduate database courses.

Rishe's book is not suitable for systems

Nothing is basically wrong with this book, except that it does not fill any useful niche.

analysts and software engineers. It does not discuss any fundamental concepts of systems analysis and design. It does not mention the lifecycle approach, dataflow diagrams, structure charts, Jackson's systems development and program design methods, entity-relationship modeling, or any other popular methods or techniques for systems analysis and design.

In the software engineering area, the Pascal-based database programming language used by Rishe exemplifies the object-oriented programming paradigm. However, he does not discuss related issues such as inheritance and persistence. The book's scope does not include other software engineering topics, such as software tools, testing and implementation, fault tolerance, structured walkthroughs, quality assurance, etc.

Rishe chose a semantic binary model to represent a conceptual database schema. Although binary models represent simple real-world objects (facts) well, they do not

differentiate between classes of objects. No clear distinction exists between values, attributes, and entities. The domain of a relation in Rishe's model can be an attribute, any group of attributes, or an entity. Its range can be a value, an attribute, a group of attributes, or an entity.

When the time comes to convert the semantic binary representation to a logical schema, it is too late to investigate what structure to impose on an object. Should the object be modeled as an entity, an attribute, or a value? As an overused example states, "color=blue" is usually a value, but it could be an attribute or even an entity in a paint factory.

Rishe addresses few difficult modeling problems. He allows generalization hierarchies but does not discuss the crucial issue of attribute inheritance. He does not challenge recursive relationships, such as bill-of-materials or course prerequisites, on either data-definition or data-manipulation levels. He only briefly mentions relationships of degree higher than two. In the network database context, he does not address the advantages of group attributes and fork (multimember) set types.

The author's choice of languages is unfortunate. The declarative language is based on predicate calculus and requires some background in mathematical logic. The programming language is based on Pascal, perhaps the least popular host language normally supported by DBMSs. Standard SQL (or even relational algebra) would be more attractive as a declarative language, and Cobol, PL/I, or C would be a better host language.

The Pascal-based language replaces the call interfaces typical of host languages with extensions to Pascal to perform data-manipulation operations. This could divert the book from a text to a monograph. In fact, the book would be an interesting monograph on database programming languages if it had sufficient depth, and if all trivialized examples were replaced with a more sophisticated discussion. In particular, Rishe should then include

references and comparisons between his Pascal-based language and such languages and environments as Adaplex, Galileo, Gemstone, KL-One, Taxis, Pascal/R, PS-Algol, and Amber.

The diagrams representing binary relations are counterintuitive and do not adhere to common practices. For example, arrows indicate a "from-to" relation rather than the type of relation (1:m or m:n). Hence, if the relation "works-in" from instructor to department is m:n, then it is not clear why only one arrow is attached to department (why can't we also say that department "employs" instructor?).

Rishe contrasts his method with normalization, which he refers to as an alternative, "bottom-up" methodology (as opposed to the "top-down" method adopted in the book). While I am not aware of any practical database design methodology that is manifestly top-down (or structured), I object to his labels for normalization. Logical database designs employ normalization as a response to particular problems caused by relational database assumptions. Normalization also has some positive effects that extend beyond relational theory, such as minimization of redundancy and reduction in update anomalies. It is not a complete design methodology. Two well-known approaches to normalization—synthesis and decomposition—attack the same problem from opposite ends. Synthesis begins with functional dependencies; decomposition departs from a universal relation. Despite this, I would not label synthesis a "bottom-up" approach, nor decomposition "top-down." In any case, Rishe seems to forget about the less-popular decomposition and does not make a fair case for normalization.

Rishe's treatment of network databases contains some simplifications and mistakes. For example, the pointers used to implement Codasyl sets should constitute rings, not chains. Therefore, if "next" pointers are used, the pointer in the last member record of a set points to the owner record (that is, the starting record of the ring). Also, the network model cannot possibly be treated as a subcategory of the binary semantic model.

The book does not contain explicit references to other works, but provides a bibliographical list. The list contains many double entries due to inconsistent spelling of some authors' names.

Most of the book seems to be the author's research work in disguise. Even if this can be defended or understood, Rishe cannot possibly justify the book's title, as it has little to do with the content. Potential buyers are warned that the book will not be a learning exercise in database design.

Leszek A. Maciaszek
University of Wollongong, Australia

Steve Jobs: The Journey is the Reward

Jeffrey S. Young (Scott, Foresman and Co., Glenview, Ill., 1988, 440 pp., \$18.95)

Most famous people are quite ordinary in most respects. Here is a chance to meet one who is not. After reading this disturbing portrayal, many adjectives come to mind to describe Steve Jobs, but "ordinary" is definitely not one of them. The company he helped create is not ordinary either.

From a garage-based operation in 1976, Apple went public in 1980 and made multimillionaires out of 24-year-olds. At a time when IBM compatibility seemed the only way that personal computer manufacturers could ensure survival, Steve Jobs pushed Apple to pursue its own path and succeeded. For some, Apple seemed the last hope for diversity in the personal computer marketplace. Jobs carefully

It's easier to see how people came to despise Jobs than to understand why he had a loyal cadre of followers.

cultivated that image, making his own name and Apple Computer into household words. Even after leaving Apple, he continues to intrigue the press, which follows Jobs' moves as he shapes his NeXT baby.

I began reading Jobs' biography with considerable interest. My initial reaction as I learned the details of his early life was that I was in for 420 pages of adulation and hero worship. The thought crossed my mind that this book might be a sanitized biography. But things got kinky as his fortunes rose, and his unique style and personality began to emerge. Readers will reflect upon those early experiences and wonder what shaped his adult behavior.

Maybe young Steve's personality took a strange twist when he stuck a bobby pin in the electric outlet, or when he and a friend ate ant poison. Perhaps the fact that his adopted father was a "repo man" had something to do with it. Or maybe Steve's parents bribed him with too many Little Richard records in his formative years to keep him quiet at four in the morning. As you read about how Jobs' true character found grander stages on which to reveal itself, and how his power made him increasingly difficult to constrain (a euphemism for "arrogant"?), you really

begin to wonder what drives this person. The contrast between public persona and private reality is remarkable.

Author Jeffrey Young was a founder of the magazine *MacWorld*. Although published independently of Apple, *MacWorld* was part of a carefully orchestrated marketing campaign arranged by Jobs and his cohorts to make the introduction of the Macintosh a media event. Young was invited to hang out with the Macintosh development team, giving him an extraordinarily intimate view of the development process and of Jobs himself. Jobs was building a group of media personalities from the ranks of the Macintosh development team, and *MacWorld* was a primary propaganda organ. The candid interwoven key players and Young's access to internal Apple memoranda provide a unique view not only of Jobs, but also of Apple's growing pains.

Young paints Jobs as a remarkably egocentric individual prone to highly erratic behavior. It's amazing that Apple survived Jobs, yet the book makes it clear that Apple as we know it today would never have existed without him. Apple walked a tightrope during his tenure. The traits that made Jobs the antithesis of the stable corporate leader also made him capable of forcing Apple in directions that secured its future, but that might just as easily have destroyed it.

I was often perplexed as I read about Jobs' treatment of the people who worked for and often cared about him. For example, he refused to provide child support while his daughter Lisa and her mother lived in a slum, although his net worth at the time was estimated at \$250 million. Yet while he disputed his paternity, he named one of Apple's computers after the girl.

Jobs agenda was not a mystery to the people who worked for him. Young recounts a story about Bill Atkinson, a key Macintosh team member whose QuickDraw graphics routines provided a foundation for the Macintosh interface. The whole project relied on Atkinson's efforts. He hit an impasse while working on a dynamic data structure known as "regions" and the tools for manipulating them. He made a breakthrough after a marathon session wrestling with the problem, but he dozed off while driving to Apple's headquarters in the middle of the night and crashed his sports car. When Atkinson regained consciousness in the hospital, Jobs was sitting by his bedside and asked him if he was okay. Atkinson said, "Don't worry, Steve. I can still remember how to do regions."

The description of how Jobs built an elite development team within Apple to create the Macintosh must be read to be believed. How many companies have an

official job description for an "evangelist" or manage to make media celebrities out of software developers? Jobs put a refrigerator in the building housing his stable of programmers and stocked it with enough mineral waters and fresh juices to cost \$100,000 a year. He surrounded the team with expensive toys like BMW motorcycles while he rubbed the noses of Apple's other divisions in the differential treatment.

(I once heard a joke that supposedly circulated in Apple, expressing the degree to which Jobs cultivated the image of an elite Macintosh club while alienating the rest of the company: How many Macintosh team members does it take to screw in a light bulb? One. He holds up the light bulb and the world revolves around him.)

At the same time, unknown to the Macintosh team, Jobs rewarded his beloved group with lower pay than comparable employees in other parts of the company. The books unfolds a set of puzzling episodes that reveal deep contradictions in Jobs' personality. Unfortunately, it's easier to see how people came to despise Jobs than to understand why he had a loyal cadre of followers able to create such innovative and excellent products.

Apple survived despite many significant mistakes that Young often attributes to Jobs. In more competitive times, Apple would not have had the chance to make as many errors. But the overall momentum of the personal computer industry, the popularity of the original Apple II, and the good fortune of VisiCalc's being first offered on the Apple II were great enough to sustain the company when other of its products were faltering.

By the mid-1980s, however, competition was keener and sales were slower. Jobs' antics were tolerated when the company was growing and reporting record profits, but when Apple reported its first quarterly loss and the Apple II computer was still the only successful product the company had brought to market, Jobs' days were numbered. The bizarre sequence of events that led to his ouster from the company he founded is not so much a story of skillful infighting and boardroom politics as a series of brawls. I found the book quite suspenseful, even though I knew the eventual outcome of Jobs' battle with CEO John Sculley.

Steve Jobs: The Journey is the Reward is not an overly charitable treatment of either Steve Jobs or the growth of Apple. It made me ponder what people will put up with if they desire a leader's praise. It made me remember the power of a demagogue able to invoke the common myths of a group. Whether you have followed the history of Apple or not, this is an interesting story about an intriguing,

if not especially sympathetic, person catapulted into a position of wealth and power.

Apple enthusiasts and Macintosh users who have witnessed events from the outside will find the book particularly interesting. If you bought into the myths in which Jobs' cloaked himself and his projects, then this book should be very

educational. It's another lesson in the power of image and propaganda. Others will find the world view of Jobs and his Macintosh development team a bit more mystifying, but still very interesting and enjoyable reading.

Gordon Goodman
Rochester Institute of Technology

Database Computers: Principles, Architectures, and Techniques

Stanley Y.W. Su (McGraw-Hill, New York, 1988, 497 pp., \$52.95)

Many organizations depend totally on the ability of their computers to accurately and quickly provide and maintain high-quality data in their databases. It follows that the performance of their chosen database management system is crucial.

The mainstream of computer development has involved computers that are

discusses the theory and experimental systems that address such bottlenecks.

In Su's terms, a database computer can comprise anything from an intelligent disk or channel to a dedicated processor system. He offers five categories (with subcategories) of database computers:

- intelligent secondary devices,
- database filters,
- associative memory systems,
- multiprocessor database computers, and
- text processors.

Each category addresses a category of bottleneck. The categories are useful, even though they overlap quite a bit.

Intelligent secondary devices address the problems of rapid storage, retrieval, and organization of data in storage. The most serious limitations of disk storage include the fact that data must be transferred out of storage to other devices for processing, that data can only be transferred one block at a time, and that data must be found by address rather than value. Intelligent storage devices address these problems by providing at least primary processing at the storage device level. Su discusses cellular logic devices and bubble memory as solutions.

Database filters reduce the I/O bottleneck in data transfers by limiting transfers to relevant or "final" data. Filtering techniques include increasing bandwidth, minimizing data to be moved by clustering based on semantic properties, and using hardware. Su is most concerned with the last method. Filters can vary greatly in capability, from selection and projection operations through joins and statistical aggregation operations. Basic filters only improve disk processing or computation to the extent they reduce the amount of data transfer. However, this reduction in data transfer may reduce the volume of computation. The more capabilities a filter has, the more it approaches the level of a back-end computer.

Associative memory enhances performance while searching by using content- and context-addressing capabilities. The data to be searched resides in the

Database Computers is one of the first books to cover the theory and implementation of such machines.

good at lots of different things—handling communications, dealing with various forms of files and access methods, providing shared use, controlling programming structures, etc. Doing different things well requires trade-offs; a general-purpose computer might do many things reasonably well, but cannot do any one thing incredibly well without giving up performance in other areas.

Users with heavy numeric-computation requirements have known this for a long time and have developed specialized processors to help their work. These processors do computations extremely well, but little else. Examples range from microcomputer coprocessors (such as the Intel 80287) to numerically oriented machines (such as the Cray).

In recent years DBMS developers and theorists have been working on similar specialized computing devices for data management. *Database Computers* is one of the first books to cover the theory and implementation of such machines.

The book begins appropriately with a discussion of DBMS computing requirements, a convincing argument for specialized processing, an in-depth analysis of the architecture and functionality of typical DBMSs, and the performance bottlenecks that arise. The text then

content-addressable memory. The search key is sent to all elements simultaneously, and those that find matches return a success signal. Thus, a search essentially takes one operation cycle regardless of the amount of data to be searched.

Several variations exist, including fully parallel searches that simultaneously compare all bits in memory; bit-serial searches that divide memory into slices, with all bits in a slice done at the same time but the slices done serially; word-serial searches that divide memory into slices corresponding to the bits within each word; and block-oriented searches that simultaneously compare parallel locations within each block of memory. The power of associative memory lies in its rapid searching ability, but it carries the disadvantages of complexity and high cost.

Multiprocessor database computers—the "top of the line"—divide into two main categories: systems with replicated functions and systems with distributed functions. The former resemble general-purpose multiprocessors except that they are organized specifically for database functions. These computers depend heavily on the ability to decompose queries into parallel operations. A typical example is the Teradata DBC/1012, which has a database processor per disk, linked together and to the host system by a specialized net and controlling processors.

The second type of system contains multiple specialized processors for specific functions. Although they are organized for database work, these systems resemble mainframe computers with intelligent disk controllers. Su discusses the Britton Lee Intelligent Database Machine as an example.

The final type of database computer is the text processor. Since text processing differs significantly from business-oriented processing, Su develops a set of primitive operations to describe the operations, followed by three different approaches to implement them. In addition, he discusses General Electric's GEScan, a high-speed text scanning and retrieval system, in some detail.

Most of the book discusses theoretical systems or experimental developments. Su reviews all the commercially available database computers that I am aware of—there aren't very many—but not in much depth. The information in his reviews of commercial systems seems several years out of date, but this does not detract from the discussion of theory.

In each section, he develops an abstract model for a type of function (such as filters) and provides algorithms for processing and for analyzing cost. Both should be of great use to anyone

involved in the design of database computers.

The text is reasonably technical, and the discussions of algorithms and performance include quite a bit of math. Even so, any reasonably knowledgeable database designer or programmer should

Speech Communication: Human and Machine

Douglas O'Shaughnessy (Addison-Wesley, Reading, Mass., 1987, 568 pp., \$49.50)

As a computer scientist working in an engineering environment, I have worked on software in such diverse areas as radar, sonar, and nuclear weapons. Just as in a business environment, a software requirements analyst in these fields must communicate with and understand the needs of the user. A programmer's ability to write software that meets the requirements in an engineering discipline often depends on how quickly he or she can get "up to speed" in that field. My bookshelf is lined with classic introductory texts: *Underwater Sound* by Urick, *Introduction to Radar* by Skolnik, *Digital Signal Processing* by Oppenheim and Schaffer, etc. *Speech Communications: Human and Machine* is another great introductory text. While this is not the first text I acquired on the subject, it should have been.

Speech Communications is excellent as an introductory text for electrical engineers at either the graduate or fourth-year undergraduate level. It is also necessary for any person who must get up to speed on the subject. The author assumes the reader has little prior knowledge of speech and presents much of the background material needed. My only complaint about the book concerns its presentation of the background material. The author's treatment of the many topics comprising Chapter 2's review of mathematics for speech processing is too cavalier. A statement that the author assumes the reader has completed courses in digital signal processing, linear algebra, and probability and statistics would suffice.

Speech Communications really begins in Chapter 3 with the basics of how the human anatomy produces speech and how speech is phonetically divided. The examples and explanations of speech production are particularly fascinating; I caught myself (actually, my wife caught me) reading the examples out loud. Chapters 4 and 5 deal with how we hear and our auditory perception. These three

grasp the contents. This book is valuable not only to designers but also to users, as it provides a basis for understanding and using the power of database computers.

Robert M. Gross
Day Data Systems

chapters, which really offer an introduction to speech, lay the groundwork for the rest of the book and provide a basic understanding of the terminology. The chapters are jammed full of information, definitions, and explanations. The author's terse writing style can make the reading difficult at times because of the quantity of information presented in such a short space. However, his style also minimizes the amount of reading time required.

I tested the book's index by writing down 20 or so terms that I came across in these chapters; the definition of each was easily found. The amount of information, the index, and the list of references make this book a handy reference.

Chapters 6-10 examine engineering-oriented aspects of speech, such as speech analysis, coding of speech signals, linear predictive coding, speech synthesis, and speech and speaker recognition. These chapters deal more with concepts than definitions, so they contain more examples, graphs, and equations. The author adopts a tutorial writing style and presents his ideas clearly and logically. The chapters on linear predictive coding and speech recognition stand out as being a bit more in depth than the rest of the book, while the chapter on speech synthesis is a bit more shallow.

Speech Communications covers a very broad field of knowledge. While the book covers all aspects of speech in sufficient detail for an introductory text, it does not cover some areas in enough depth for all readers. For those needing more information, I recommend the references given in the text. The references are fairly complete and run through 1986.

Speech Communications is a welcome addition to my library and I recommend it. However, be sure to skim it first. Some areas of particular interest to you might not be adequately covered, and some background knowledge is required.

Jeff Cartwright
BDM Corp.

CALL FOR PAPERS

The Second IEEE Symposium on Computer-Based Medical Systems



The Computer Society of the IEEE



Engineering in Medicine and
Biology Society of the IEEE

UNIVERSITY OF MINNESOTA

June 25-27, 1989 • Minneapolis, Minnesota

The Program

CBMS combines papers, presentations, discussions and panels. Both contributed and invited papers will be presented, assuring an excellent and balanced program. All papers accepted and that meet the deadline for final copy will be published in the Symposium Proceedings. There will be four tracks related to Computer-Based Medical Systems:

- Clinical Assessment and Risk Evaluation (New)
- Regulations of and Standards for CBMS
- Reliability and Validity of CBMS
- Artificial Intelligence and Neural Networks

Schedule for Contributed Papers

Call for Papers	October
Papers Due	January 10, 1989
Notice of Acceptance	February 10, 1989
Camera ready papers due	March 10, 1989

Papers should include (1) the title of the paper; (2) names of the authors and (3) name, title, address and telephone number of the primary author. They should be typed, double-spaced. Send four copies to Dr. Galle, Continuing Medical Education, University of Minnesota, Box 202 UMHC, 420 Delaware Street, S.E., Minneapolis, MN 55455, USA, (612) 626-5525

Who Should Attend

This conference is intended for engineers and computer scientists from industry who are designing and developing Computer-Based Medical Systems (CBMS). Reports about applications in progress are encouraged. Academic and professional people who wish to discuss and define their needs for CBMS or who are working on such systems will also find the Symposium valuable. The Symposium provides an excellent opportunity for government regulators to interact with those working in the CBMS area. Medical residents and fellows working on computer projects and advanced engineering and computer science students working on medical projects are especially encouraged to submit papers describing their work. We anticipate an award for the best student paper(s).

Information

If you can assist with the program or have suggestions for topics or speakers call or write Dr. Kriewall. All other correspondence should be directed to Dr. Galle of: Continuing Medical Education, University of Minnesota, Box 202 UMHC, 420 Delaware Street, S.E., Minneapolis, MN 55455, USA, (612) 626-5525

Host Cities

The Minneapolis and St. Paul area is home for several hundred medical high-tech companies, almost all of which have products that are computer-based. The Twin Cities of Minneapolis and St. Paul are also rich in culture and heritage. Minneapolis has long been known as the City of Lakes. St. Paul is the seat of Minnesota's state government.

Major league sports moved to downtown Minneapolis with the completion of the Hubert H. Humphrey Metrodome in 1982. The "Dome" is home to the Minnesota Vikings football team and the 1987 World Series Baseball Champions, the Minnesota Twins.

Other area interests include the nationally recognized Minnesota Orchestra, St. Paul Chamber Orchestra, and the Guthrie Theater, as well as the technologically-advanced Science Museum of Minnesota / William L. McKnight-3M Omnitheater. There are other acclaimed museums including the Minneapolis Institute, the Walker and the new Sculpture Garden.

PLAN NOW TO PARTICIPATE!

Symposium Committee

PROGRAM CHAIRMAN

Timothy J. Kriewall, Ph.D.
Sarns, Inc./3M
6200 Jackson Rd.
Ann Arbor, MI 48103
USA
(313) 663-4145

EXECUTIVE COMMITTEE

Bart Galle, Ph.D.
Continuing Medical Education
University of Minnesota

James E. Holte, Ph.D.
Department of Electrical
Engineering
University of Minnesota
(Representing IEEE E/IBS)

John M. Long, Ed.D.
Department of Surgery
University of Minnesota

H. Troy Nagle, M.D., Ph.D.
North Carolina State University
Raleigh, NC
(Representing IEEE CS)

Al Potvin, Ph.D.
Eli Lilly & Company
Indianapolis, IN
(Representing IEEE E/IBS)

COOPERATING ORGANIZATIONS

Region 7 IEEE
Twin Cities Chapters of CS, E/IBS
IEEE CS Technical Committee on
Computational Medicine

ADVERTISER INDEX

Academic Press, Inc.	5
CACI Products Company	1
Compsac '89	100
Computer Arithmetic Symposium	87
Computer-Based Medical Systems	119
Computer Society Membership	24A-25, 103
Computer Society Ombudsman	6
Data Engineering Conference	58-59
Digital Equipment Corp.	67
EDS Research and Development	99
The George Washington University	93
Hypercube Concurrent Computers and Applications	85
ICCD '89	90
IEEE Transactions on Computers	97
IEEE Transactions on Knowledge and Data Engineering	88A-B
Index Technology Corp.	8-9
Information Research Associates	Cover II
Institute for Defense Analyses	55
King Fahd University of Petroleum & Minerals	37
Meta Software Corp.	Cover IV
Motorola Inc., Cellular Group	45
Nanyang Technological Institute	81
New Jersey Institute of Technology	93, 99
Software Maintenance Conference	94
U.S. General Accounting Office	56
Visible Systems Corp.	7
Classified Advertising	104-114

FOR DISPLAY ADVERTISING INFORMATION, CONTACT:

Northern California and Pacific Northwest: Roy McDonald Assoc. Inc., 5915 Hollis St., Emeryville, CA 94608; (415) 653-2122.

Jim Olsen, P.O. Box 696, Hillsboro, OR 97123; (503) 640-2011.

Southern California and Mountain States: Richard C. Faust Co., 24050 Madison St., Suite 100, Torrance, CA 90505; (213) 373-9604.

Southwest: The House Co., 3000 Wesleyan, Suite 345, Houston, TX 77027; (713) 622-2868.

Midwest: Robert Acker & Assoc., 480 Central Ave., Northfield, IL 60093; (312) 441-6050.

East Coast: Atlantic Representative Group, 349 Maple Pl., Keyport, NJ 07735; (201) 739-1444.

New England: Arpin Associates, PO Box 227, Weston, MA 02193; (617) 899-5613. Paul Gillespie, 40 Sterling St., Somerville, MA 02144; (617) 625-1777.

George W. Watts, III, 6 Ladyship Cir., Amherst, MA, MA 01002; (413) 253-9881.

Europe: Heinz J. Gorgens, Parkstrasse 8a, D-4054 Nettetal 1 - Hinsbeck (F.R.G.); phone: (0 21 53) 8 99 88; telex 841(17)2153310-HJG tix d.

Advertising Director: Dawn Peck. For production information, conference, and classified advertising, contact Heidi Rex or Marian Tibayan.

COMPUTER, 10662 Los Vaqueros Cir., Los Alamitos, CA 90720; phone (714) 821-8380; fax (714) 821-4010.

PRODUCT INDEX

	RS#	Page#
BOARDS		
Real-time boards	42-44	84
CIRCUITS		
Floating-point coprocessor	53-54	86
IC announcements	120-132	88
COMPONENTS		
Microsystem announcements	135-148	89
CONFERENCES & COURSES		
Compsac '89	—	100
Computer Arithmetic	—	87
Computer-Based Medical Sys.	—	119
Data Engineering	—	58-59
Hypercube Concurrent Computers	—	85
ICCD '89	—	90
Software Maintenance	—	94
DATA COMMUNICATIONS		
Graphical interface	1	C.II
Network software	45	84
PUBLISHERS		
CASE management	4	8-9
CS membership	—	24A-25, 97, 103
CS ombudsman	—	6
Knowledge and data engineering	—	88A-B
Parallel and distributed computing	2	5
RECRUITMENT		
Classified advertising	—	104-114
Technical professional	37, 45, 55, 56, 67, 81, 93, 99	
SOFTWARE		
CAE/CAD software	30-31	82
SYSTEMS		
Computer	32-33, 34, 35-36, 46-49, 50-51	82-84, 86
Development environment	40-41	83
Expert system tool	21-25	69-81
Graphics development shell	5	C.IV
Project management	52	86
Productivity tool	3	7
Simulation package	—	1
Workstation	37-38	83
TEST & MEASUREMENT EQUIPMENT		
Distributed array processor	39	83

COMPUTER

November 1988 issue (Card void after April 1989)

Please print or use peel-off address label.

Name _____

Company _____

Address _____

City _____

State/ZIP _____

Country _____

Please send (circle those you want)

201 Publications catalog	203 Student membership information
202 Membership information	204 Application for senior membership

Reader interest

(Circle what you liked;
add comments on the back)

- A Programmer's Apprentice
B Reducing Contention
C Using Inspections
D Taxonomy
- E President's Message
F EIC's Message
G Letters
H Standards
J Computer Society News
K Update
L New Product Reviews
M New Products
N IC Announcements
P Microsystems
Q Conferences
R Calendar
S Call for Papers
T Book Reviews

Product inquiries

(circle numbers for products and advertisements
on which you want more information)

1	21	41	61	81	101	121	141	161	181
2	22	42	62	82	102	122	142	162	182
3	23	43	63	83	103	123	143	163	183
4	24	44	64	84	104	124	144	164	184
5	25	45	65	85	105	125	145	165	185
6	26	46	66	86	106	126	146	166	186
7	27	47	67	87	107	127	147	167	187
8	28	48	68	88	108	128	148	168	188
9	29	49	69	89	109	129	149	169	189
10	30	50	70	90	110	130	150	170	190
11	31	51	71	91	111	131	151	171	191
12	32	52	72	92	112	132	152	172	192
13	33	53	73	93	113	133	153	173	193
14	34	54	74	94	114	134	154	174	194
15	35	55	75	95	115	135	155	175	195
16	36	56	76	96	116	136	156	176	196
17	37	57	77	97	117	137	157	177	197
18	38	58	78	98	118	138	158	178	198
19	39	59	79	99	119	139	159	179	199
20	40	60	80	100	120	140	160	180	200

1

COMPUTER

November 1988 issue (Card void after April 1989)

Please print or use peel-off address label.

Name _____

Company _____

Address _____

City _____

State/ZIP _____

Country _____

Please send (circle those you want)

201 Publications catalog	203 Student membership information
202 Membership information	204 Application for senior membership

Reader interest

(Circle what you liked;
add comments on the back)

- A Programmer's Apprentice
B Reducing Contention
C Using Inspections
D Taxonomy
- E President's Message
F EIC's Message
G Letters
H Standards
J Computer Society News
K Update
L New Product Reviews
M New Products
N IC Announcements
P Microsystems
Q Conferences
R Calendar
S Call for Papers
T Book Reviews

Product inquiries

(circle numbers for products and advertisements
on which you want more information)

1	21	41	61	81	101	121	141	161	181
2	22	42	62	82	102	122	142	162	182
3	23	43	63	83	103	123	143	163	183
4	24	44	64	84	104	124	144	164	184
5	25	45	65	85	105	125	145	165	185
6	26	46	66	86	106	126	146	166	186
7	27	47	67	87	107	127	147	167	187
8	28	48	68	88	108	128	148	168	188
9	29	49	69	89	109	129	149	169	189
10	30	50	70	90	110	130	150	170	190
11	31	51	71	91	111	131	151	171	191
12	32	52	72	92	112	132	152	172	192
13	33	53	73	93	113	133	153	173	193
14	34	54	74	94	114	134	154	174	194
15	35	55	75	95	115	135	155	175	195
16	36	56	76	96	116	136	156	176	196
17	37	57	77	97	117	137	157	177	197
18	38	58	78	98	118	138	158	178	198
19	39	59	79	99	119	139	159	179	199
20	40	60	80	100	120	140	160	180	200

2

COMPUTER

November 1988 issue (Card void after April 1989)

Please print or use peel-off address label.

Name _____

Company _____

Address _____

City _____

State/ZIP _____

Country _____

Please send (circle those you want)

201 Publications catalog	203 Student membership information
202 Membership information	204 Application for senior membership

Reader interest

(Circle what you liked;
add comments on the back)

- A Programmer's Apprentice
B Reducing Contention
C Using Inspections
D Taxonomy
- E President's Message
F EIC's Message
G Letters
H Standards
J Computer Society News
K Update
L New Product Reviews
M New Products
N IC Announcements
P Microsystems
Q Conferences
R Calendar
S Call for Papers
T Book Reviews

Product inquiries

(circle numbers for products and advertisements
on which you want more information)

1	21	41	61	81	101	121	141	161	181
2	22	42	62	82	102	122	142	162	182
3	23	43	63	83	103	123	143	163	183
4	24	44	64	84	104	124	144	164	184
5	25	45	65	85	105	125	145	165	185
6	26	46	66	86	106	126	146	166	186
7	27	47	67	87	107	127	147	167	187
8	28	48	68	88	108	128	148	168	188
9	29	49	69	89	109	129	149	169	189
10	30	50	70	90	110	130	150	170	190
11	31	51	71	91	111	131	151	171	191
12	32	52	72	92	112	132	152	172	192
13	33	53	73	93	113	133	153	173	193
14	34	54	74	94	114	134	154	174	194
15	35	55	75	95	115	135	155	175	195
16	36	56	76	96	116	136	156	176	196
17	37	57	77	97	117	137	157	177	197
18	38	58	78	98	118	138	158	178	198
19	39	59	79	99	119	139	159	179	199
20	40	60	80	100	120	140	160	180	200

3

Editorial comments

I liked: _____

I disliked: _____

I would like to see: _____

For reader service inquiries, use other side

PO box address for
reader service cards only

PLACE
STAMP
HERE

**Computer
Reader Service Inquiries
PO Box 16508
North Hollywood, CA 91615-6508
USA**

Editorial comments

I liked: _____

I disliked: _____

I would like to see: _____

For reader service inquiries, use other side

PO box address for
reader service cards only

PLACE
STAMP
HERE

**Computer
Reader Service Inquiries
PO Box 16508
North Hollywood, CA 91615-6508
USA**

Editorial comments

I liked: _____

I disliked: _____

I would like to see: _____

For reader service inquiries, use other side

PO box address for
reader service cards only

PLACE
STAMP
HERE

**Computer
Reader Service Inquiries
PO Box 16508
North Hollywood, CA 91615-6508
USA**



IEEE COMPUTER SOCIETY

A member society of the Institute of Electrical and Electronics Engineers, Inc.

Executive Committee

President: Edward A. Parrish, Jr.
Vanderbilt University
School of Engineering
Nashville, TN 37235
(615) 322-2762

President-Elect: Kenneth R. Anderson*

Vice Presidents

Standards: Helen M. Wood (1st VP)*

Conferences and Tutorials: Joseph E. Urban (2nd VP)*

Area Activities: Willis King†

Computer Society Press: Duncan H. Lawrie

Education: Gerald L. Engelt

Membership and Information: Merlin G. Smith†

Publications: James H. Aylor†

Technical Activities: Laurel V. Kaleda

Secretary: Duncan H. Lawrie

Treasurer: Barry W. Johnson†

Past President: Roy L. Russo*

Division V Director: Harriett Rigast

Division VIII Director: H. Troy Nagle, Jr.†

Executive Director: T. Michael Elliott†

*voting member of the Board of Governors

†nonvoting member of the Board of Governors

Board of Governors

Term Ending 1988

Mario Barbacci
Victor Basil
Paul Borrill
Lorraine M. Duval
Michael Evangelist
Allen L. Hankinson
Laurel V. Kaleda
Ted Lewis
Ming T. Liw
Earl E. Schwartzlander, Jr.

Term Ending 1989

Bill D. Carroll
Lansing Hatfield
Duncan H. Lawrie
David Pessel
Susan L. Rosenbaum
Sallie W. Sheppard
Bruce D. Shriver
Harold S. Stone
Akikiko Yamada
Marshall C. Yovits

Next Board Meeting

March 3, 1989: 8:30 a.m.
Cathedral Hill Hotel, San Francisco, CA

Senior Staff

Executive Director: T. Michael Elliott
Editor and Publisher: True Seaborn
Director, Computer Society Press: Chip G. Stockton
Director, Conferences: Anna Marie Kelley
Director, Finance and Administration: Tod S. Heisler

Computer Society Offices

Headquarters Office

1730 Massachusetts Ave. NW
Washington, DC 20036-1903
Phone: (202) 371-0101
Telex: 7108250437 IEEE COMPSO

Publications Office

10662 Los Vaqueros Circle
Los Alamitos, CA 90720

Membership and General Information: (714) 821-8380

Publications Orders: (800) 722-6657

European Office

13, Avenue de l'Aquilon
B-1200 Brussels, Belgium
Phone: 32 (2) 770-21-98
Telex: 25387 AVVALB

Asian Office

Ooshima Building
2-19-1 Minami-Aoyama, Minato-ku
Tokyo 107, Japan
Phone: 81 (3) 408-3553

Use the Reader Service Card to obtain the following information:

- Membership application—student #203, others #202
- Periodicals subscription form for individuals #200
- Periodicals subscription form for organizations #199
- Publications catalog #201
- Standards working groups list #195
- COMPMail+ international electronic mail/database brochure #194
- Technical committee list/application #197
- Chapters lists, start-up procedures—student/regular #193
- Student scholarship information #192
- Awards description/nomination forms #198
- Volunteer leaders/staff directory #196
- IEEE senior member application #204

Purpose

The Computer Society advances the theory and practice of computer science and engineering, promotes the exchange of technical information among 97,000 members worldwide, and provides a wide range of services to members and nonmembers.

Membership

Members receive the acclaimed monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others seriously interested in the computer field.

Publications and Activities

Computer. An authoritative, easy-to-read magazine containing tutorial and in-depth articles on topics across the computer field; plus news, conferences, calendar, interviews, new products, etc.

Periodicals. The society publishes six magazines and three research transactions. Refer to membership application or request information as noted above.

Conference Proceedings, Tutorial Texts, Standards Documents. The Computer Society Press publishes more than 100 titles every year.

Standards Working Groups. Over 100 of these groups produce IEEE standards used throughout the industrial world.

Technical Committees. Over 30 TCs provide newsletters, interaction with peers in specialty areas, and have direct influence on standards, conferences, education, etc.

Conferences/Education. The society holds about 100 conferences each year and sponsors many educational activities, including computing sciences accreditation.

Chapters. Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

European Office

Payments for Computer Society membership and publication orders are accepted by cheques in Belgian, British, German, Swiss, or U.S. currency, or by American Express, Eurocard, MasterCard, or Visa credit cards.

Ombudsman

Members experiencing problems—magazine delivery, membership status, unresolved complaints—may write to the ombudsman at the Publications Office.



Honeywell-Bull used Design/OA to create a graphical micro-computer interface to monitor their nightly mainframe batch processing. The application retrieves batch processing data, initiates the batch jobs, and provides the operator real-time processing status throughout the night.

Planning Research Corporation used Design/OA to develop an Ada diagramming tool by adding Ada specific graphics to Design/2.0. This new tool, called AdaDesign, is used by Ada programmers for flow charting and system design.

"With Design/QA we can go from an idea to an implementation very quickly. It has expanded the range of questions we can explore, given the same resources."

Alexander Levis, Senior Research Scientist, Laboratory for Information and Decision Systems.

AT&T Information Systems used Design/DA to build a reverse engineering application that provides dynamic documentation of large mainframe C programs. Programmers can graphically view the program's functions, files, variables, and their relationships.

Design/OA, The Open Architecture Development System from Meta Software, is the graphics development shell that enables you to create graphic applications in a fraction of the time required by other methods.

Design/OA is used to build tools for software engineering, system design, structured analysis, and system simulation. Virtually any system design methodology can be represented graphically with Design/OA. Design/OA is also an effective means of creating graphic interfaces to other micro, mini or mainframe applications and databases.

Complex graphic relationships can be rapidly defined and manipulated, both in overview and across lower levels of detail. You need only work with high-level support tools, enhancing development productivity and minimizing implementation time, without changing your development style. These tools also make it easy to modify an application.

In addition to supporting developers worldwide, Design/QA is the foundation for Meta Software's family of system modeling tools. Meta Software offers Design/QA users both initial training support and advanced implementation services.

To explore the potential of Design/OA for your application, write Andrew Levin, Meta Software Corp., 150 CambridgePark Drive, Cambridge, MA 02140, or call 800-227-4106 (617-576-6920 in MA).



Meta Software